FIPS PUB 73

FEDERAL INFORMATION
PROCESSING STANDARDS PUBLICATION

1980 JUNE 30

U.S. DEPARTMENT OF COMMERCE / National Bureau of Standards

GUIDELINES FOR SECURITY OF COMPUTER APPLICATIONS

CATEGORY: ADP OPERATIONS
SUBCATEGORY: COMPUTER SECURITY

U.S. DEPARTMENT OF COMMERCE, Philip M. Klutznick, *Secretary*
Jordan J. Baruch, *Assistant Secretary for Productivity, Technology and Innovation*
NATIONAL BUREAU OF STANDARDS, Ernest Ambler, *Director*

Foreword

The Federal Information Processing Standards Publication Series of the National Bureau of Standards is the official publication relating to standards adopted and promulgated under the provisions of Public Law 89-306 (Brooks Act) and under Part 6 of Title 15, Code of Federal Regulations. These legislative and executive mandates have given the Secretary of Commerce important responsibilities for improving the utilization and management of computers and automatic data processing systems in the Federal Government. To carry out the Secretary's responsibilities, the NBS, through its Institute of Computer Sciences and Technology, provides leadership, technical guidance and coordination of Government efforts in the development of guidelines and standards in these areas.

As every facet of the Federal Government becomes increasingly dependent upon ADP systems, concern about the protection, availability, and reliability of Federal agency data and computer applications has become evident in the executive and legislative branches of the Government as well as in the minds of individual citizens. This guideline was developed, as part of an overall Department of Commerce computer security and risk management program, to provide technical and managerial guidance to Federal agencies that will enable them to reduce or eliminate unnecessary and excessively high risks that are associated with the utilization of automated information systems. NBS is pleased to make these Guidelines for Security of Computer Applications available for use by Federal agencies.

James H. Burrows, *Director*
Institute for Computer Sciences
and Technology

Abstract

Security decisions should be an integral part of the entire planning, development, and operation of a computer application. This guideline describes the technical and managerial decisions that should be made in order to assure that adequate controls are included in new and existing computer applications to protect them from natural and human-made hazards and to assure that critical functions are performed correctly and with no harmful side effects. The multifaceted nature of computer security is described, and differences in security objectives, sensitivity levels, and vulnerabilities that must be considered are identified. Fundamental security controls such as data validation, user identity verification, authorization, journalling, variance detection, and encryption are discussed as well as security-related decisions that should be made at each stage in the life cycle of a computer application. These include questions about security feasibility and risk assessment that should be asked during initial planning, decisions that should be made during the design, programming and testing phases, controls that should be enforced during the development process, and security provisions that should be enforced during the day-to-day operation of the system.

Key words: ADP availability; ADP security; application system security; computer applications; computer reliability; computer security; data confidentiality; data integrity; data security; Federal Information Processing Standards Publication; security controls; system life cycle; system security.

Nat. Bur. Stand. (U.S.), Fed. Info. Process. Stand. Publ. (FIPS PUB) 73, 55 pages.
(1980)
CODEN:FIPPAT

For sale by the National Technical Information Service, U.S. Department of Commerce, Springfield, Virginia 22161.

Federal Information
Processing Standards Publication 73

1980 June 30

ANNOUNCING THE

GUIDELINES FOR SECURITY OF COMPUTER APPLICATIONS

Federal Information Processing Standards Publications are issued by the National Bureau of Standards pursuant to the Federal Property and Administrative Services Act of 1949 as amended, Public Law 89-306 (79 Stat. 1127), and as implemented by Executive Order 11717 (38 FR 12315, dated May 11, 1973), and Part of Title 15 CFR (Code of Federal Regulations).

Name of Guideline: Guidelines for Security of Computer Applications

Category of Guideline: ADP Operations, Computer Security

Explanation: These guidelines describe methods and techniques that can reduce the hazards associated with computer applications. They describe the different security objectives for a computer application, explain the control measures that can be used, and identify the decisions that should be made at each stage in the life cycle of a sensitive computer application. Other publications that provide a more exhaustive treatment of specific subjects are referenced.

Approving Authority: Department of Commerce, National Bureau of Standards (Institute for Computer Sciences and Technology).

Maintenance Agency: Department of Commerce, National Bureau of Standards (Institute for Computer Sciences and Technology).

Cross Index:

a. Federal Information Processing Standards Publication (FIPS PUB) 31, Guidelines for Automatic Data Processing Physical Security and Risk Management.

b. Federal Information Processing Standards Publication (FIPS PUB) 39, Glossary for Computer System Security.

c. Federal Information Processing Standards Publication (FIPS PUB) 41, Computer Security Guidelines for Implementing the Privacy Act of 1974.

d. Federal Information Processing Standards Publication (FIPS PUB) 46, Data Encryption Standard.

e. Federal Information Processing Standards Publication (FIPS PUB) 48, Guidelines on Evaluation of Techniques for Automated Personal Identification.

f. Federal Information Processing Standards Publication (FIPS PUB) 65, Guideline for Automatic Data Processing Risk Analysis.

Applicability: These guidelines apply to the planning, development, and operation of Federal computer applications which require various degrees of protection due to the nature of the data processed or to the risk and magnitude of loss or harm that could result from improper operation or deliberate manipulation of the applications. They are written to be used by the managers of any organization that depends on, or by the data processing personnel who are involved with, such applications. They are especially applicable to the development of new applications and to the revision of existing applications. They should be followed from the time of the initial planning for the application or for its revision.

Implementation: These guidelines should be used by each organization in which new computer applications are developed and as current applications are improved, particularly when analyzing protection requirements and when selecting and implementing security safeguards commensurate with calculated risks. Depending upon differing operational requirements, applications will require various levels of security protection. These guidelines, when used in conjunction with the references at the end of this document, should assist those responsible for the computer application in making and justifying essential security decisions.

Specifications: Federal Information Processing Standards Publication (FIPS PUB) 73, Guidelines for Security of Computer Applications, (affixed).

Qualifications: These guidelines are not intended to satisfy completely the more critical security requirements for Department of Defense tactical applications or applications that process national security data. Similarly, they are not intended to give complete guidance for real-time control applications where signals to control critical processes are required within a few seconds or less (e.g., aircraft flight control, nuclear power plant control). While the guidance contained in this document is applicable to these applications, approaches that are more rigorous than those discussed in these guidelines are often required to provide adequate levels of protection.

Where to Obtain Copies of the Guideline: Copies of this publication are for sale by the National Technical Information Service, U.S. Department of Commerce, Springfield, Virginia 22161. When ordering, refer to Federal Information Processing Standards Publication 73 (FIPS-PUB-73) and title. When microfiche is desired, this should be specified. Payment may be made by check, money order, or deposit account.

Acknowledgments

The Institute of Computer Sciences and Technology acknowledges Theodore A. Linden of Xerox Corporation as a principal author of this publication, Sheila Brand of DHHS as a major contributing author, and Robert G. McKenzie of NASA, John Panagacos of Equitable Life Assurance Society, and Bruce Wellborn of GSA as contributing authors.

The Institute is grateful to those members of the Federal Information Processing Standards Task Group 15 on Computer Systems Security who contributed innovative and imaginative ideas to the development of these guidelines, which originated in that task group.

Federal Information
Processing Standards Publication 73

1980 June 30

Specifications for

GUIDELINES FOR SECURITY OF COMPUTER APPLICATIONS

CONTENTS

ACTION SUMMARY

The essential recommendations from these guidelines are summarized here to provide a quick overview of action items for improving the security of computer applications. Actions that are part of the minimum requirements for each agency's computer security program as defined in OMB Circular A-71, Transmittal Memorandum No. 1 (A-71, TM1), July 27, 1978, are identified by an asterisk. The sections of this document that give more details about the action are identified within parentheses; for example, (2) means the whole of section 2, and (2.1) means all of section 2.1.

I. Determine security objectives, the degree of sensitivity, and the vulnerabilities of the application and its data. (2)

*For applications that are already operational, begin with a risk analysis following the guidance in [FIPSP 79B]. (4.2)

For new applications, assess the risks and the feasibility of securing the application. This must be done during the feasibility studies and cost-benefit analyses that are performed before resources are committed to development of the new application. (5)

II. For existing applications, a risk analysis usually uncovers some vulnerabilities that need immediate attention. Often, improved control over the manual procedures and operational practices will reduce the risk significantly, while more thorough improvements are planned and developed. (7)

The remaining action items all apply to the revision of existing applications as well as to new applications.

* III. Define security specifications for the application. These should be developed as part of the definition of requirements for the application. (6.1)

IV. Design the interfaces of the system so that unnecessary vulnerabilities are avoided and so that the risk exposure is minimized. (6.2)

V. Include in the system design enough basic controls so the vulnerabilities that cannot be avoided are controlled and managed. (3)

* VI. Conclude the design stage with a detailed design review by a team of experts who were not part of the design effort. For any sensitive application, this design review must include an analysis of the risks that will remain in the new or revised system. (6.2.8)

VII. Plan and control the programming stage to minimize the danger that errors or deliberate traps in the programs will be a source of security exposures. (6.3)

* VIII. Test the system not only to evaluate whether the desired functionality is implemented correctly but also to determine how the system will handle unexpected or fraudulent inputs and how it will perform under other unusual conditions. (6.4)

IX. Once the system is operational, ensure that planned security procedures are followed in all the manual activities associated with the system. The careful planning for security will be wasted if management does not emphasize security during the day-to-day operation of the system. (7)

* X. Contingency plans should be developed to assure the integrity of the data processed and the continuity of the application's critical functions. The plan must be implemented (i.e., the prerequisite activities such as training of personnel, alternate site selection, selection of backup file storage site, determination of critical functions, etc. must be completed) and maintained in a state of readiness so that responses to emergencies will be timely and successful. (7.6)

1. INTRODUCTION

Computer security concerns the protection of computers and their services from all natural and human-made hazards. As computers perform more and more critical services and as effective human review becomes more difficult, the most serious security concerns often become a matter of performance assurance—an assurance that the computer performs its critical functions correctly and that there are no harmful side-effects [PATRR 78].

Computer services must be protected not only from physical threats such as damage and theft but also from internal vulnerabilities such as programming errors and misuse by authorized users [RCBCG 79]. Previous guidelines [FIPSP 74] dealt extensively with security against physical threats to the computer facility. These guidelines extend [FIPSP 74] by providing more detail on ways to control internal vulnerabilities.

The term "security" is sometimes used in a narrower sense to cover only protection against unauthorized disclosure of information. In this document, "security" has a broader meaning and includes most integrity and performance assurance concerns.

These guidelines focus on controls for use with computer applications. Inadequacies in the design and operation of computer applications [BOEHB 73A], [FIFED 77], [MCCAJ 77A], [MCCAJ 77B], [MCCAJ 77C], [YOURE 75], [MILLH 77] are a very frequent source of harmful effects associated with computers, and in most cases the effort to improve security should concentrate on the application systems. An exception occurs when the nation's security or well-being could be seriously affected. In such situations, there is a significant danger that flaws inherent in the operating system could be exploited to avoid controls built into the application system; and for these applications security should be built in from the bottom up—beginning with the hardware and the operating system. Until operating system security improves to the point that operating systems can be trusted, the majority of Federal applications must coexist with current operating systems. For these applications, vast improvements in security are usually possible before inadequacies in the operating system become a limiting factor.

Security concerns should be an integral part of the entire planning, development, and operation of a computer application. Much of what needs to be done to improve security is not clearly separable from what is needed to improve the usefulness, reliability, effectiveness, and efficiency of the computer application. These guidelines are compatible with other FIPS guidance and with best current practices for achieving these other goals. On the other hand, many tradeoffs must be made between security and other goals. The tradeoffs need not always be resolved in favor of security, but security requirements must be appropriately weighed in each tradeoff, and they cannot be consistently neglected.

Security is given special treatment in these guidelines because it is a new and difficult problem to some users and managers and it is frequently neglected until it is too late to take the most effective action. Many of the hazards associated with computer applications do not occur frequently, and most managers have not had firsthand experience with them. It is easy for managers to neglect these hazards and become enmeshed in the smaller, more immediate, and more obvious problems.

Section 2 describes the multifaceted nature of computer security and identifies the differences in objectives, sensitivity levels, and vulnerabilities that must be considered. Section 3 is more technical and describes the basic controls that are commonly applicable. Readers who are applying these guidelines to a specific application system may skim this section and then later use it as reference material on the specific topics that are useful in their application. The remainder of the document—sections 4 to 7—describe the security-related decisions that should be made at each stage in the life cycle of a computer application. Section 4 introduces the life cycle approach to security—both for new systems and for revisions to existing systems. Section 5 identifies the questions about security feasibility and risk assessment that should be asked during the initial planning for an application system. Section 6 describes the decisions about security that should be made at each stage of the system development and it describes controls that should be enforced during the development process. Section 7 deals with the security provisions to be enforced during the day-to-day operation of the system.

2. SECURITY OBJECTIVES AND VULNERABILITIES

Computer security is needed because undesirable events during computer processing can have effects such as denial of benefits due to citizens, unauthorized disclosure of sensitive information, loss of Government money or resources, human injury or illness, false arrest, and, in extreme cases, loss of life and endangerment of the national welfare and security. Since the possible effects are so different, specific security objectives must be determined for each application.

This section describes general ways of classifying security objectives and illustrates some of the differences among various applications.

2.1 Integrity, Confidentiality, and Availability

To identify objectives that are common to many diverse computer applications, it is useful to classify undesirable events not in terms of their ultimate effects but in terms of their proximate effects on computerized data. An event may result in unauthorized or undesirable

* modification or destruction of data,
* disclosure of data,
* unavailability of data or system services.

This classification of undesirable events corresponds to the following three security objectives for computer systems:

1. DATA INTEGRITY—The state that exists when computerized data is the same as that in the source documents or has been correctly computed from source data and has not been exposed to accidental or malicious alteration or destruction. Erroneous source data and fictitious additions to the data are also considered violations of data integrity.
2. DATA CONFIDENTIALITY—The state that exists when data is held in confidence and is protected from unauthorized disclosure. Misuse of data—by those authorized to use it for limited purposes—is also a violation of data confidentiality.
3. ADP AVAILABILITY—The state that exists when required ADP services can be obtained within an acceptable period of time[1].

Many controls [RUTHZ 77], [RUTHZ 78], [EDPAF 77], [IIASA 77], [SGCCA 75B], [MAIRW 76] are effective in meeting all three security objectives; however, some are specific to one objective and may even have a detrimental effect on other objectives. For example, the damage resulting from loss or destruction of data (a data integrity problem) can be reduced by keeping multiple copies of the data—but this makes data confidentiality more difficult. Thus it is important to determine how much weight must be put on each of the three objectives of data integrity, data confidentiality, and ADP availability.

2.2 Accidental and Deliberate Events

A second classification of objectives arises from the differences between accidental and deliberate acts. Accidents, errors, and omissions generally account for more losses than deliberate acts; and thus they should be the first focus of attention. Safeguards that reduce the potential for harmful effects from accidents are also a first step toward reducing the opportunities for fraud and deliberate misuse—a system that tolerates frequent errors is a fertile ground for criminal activity that can be masked by the errors.

While controls that reduce accidental failings are the first step, many systems must also be protected against deliberate fraud and misuse [BEQUA 78A], [BEQUA 78B], [ALLEB 76], [PARKD 73], [PARKD 75], [PARKD 76], [RCBCG 76B], [KRAUL 79]. For some applications, existing legal sanctions[2] along with external efforts to identify criminal activity may be adequate; however, for a growing number of applications, it is necessary to include internal controls to prevent or at least detect criminal activity within the application.

Controls against deliberate acts must be carefully thought out because deliberate acts can exploit the weakest points in the control system. Controls that do not reduce the vulnerability of the weakest points will do little to deter deliberate acts. To protect a house against accidental rain damage, each window that is closed reduces the potential damage; to protect the house against a burglar, it does little good to lock second-story windows unless *all* the weaker points on the first story are protected.

Security against deliberate acts must control all the most likely avenues of attack; however, substantial deterence against fraud and embezzlement can be achieved if a potential perpetrator believes there is a definite, however small, probability of being detected and apprehended. Many potential white-collar criminals will be deterred when it is clear that the act is illegal and that there is no way to commit the crime without some distinct chance of being detected and

[1] Equipment Security—the protection of the investment in the computer itself and in related equipment—is a fourth area of concern; however, this is usually separate from the security concerns of the application system [FIPSP 74], [RCBCG 76C].

[2] Current legal sanctions for computer-related crime are weak and are often difficult to enforce.

prosecuted. Unfortunately, even this limited goal is difficult to achieve. Computers are very predictable and they follow their programmed instructions with great regularity; these properties are very useful to a criminal in preserving the appearance of normality while committing a crime.

2.3 Examples of Sensitive Applications

The degree of sensitivity of an application depends on the nature of the data it processes and on the way the application system is used to support other activities. Each agency must determine the degree to which its computer applications are sensitive. Examples of applications that might be sensitive are given below. The groupings reflect systems that have somewhat similar security problems; they do not necessarily reflect increasing levels of sensitivity.

2.3.1 Applications Providing General Processing Support

MAJOR OBJECTIVES: data integrity

EXAMPLES:

* Engineering calculations used in aircraft design
* Query systems that support health care decisionmaking
* Automated scheduling of safety inspections
* Simulation of the dispersion of toxic substances

In these applications, accidents are a major concern. Health, safety, welfare, and even lives may depend on the correctness of the output from these applications. Thus data integrity—including integrity of the programs—is critical; however, if the application routinely produces accurate and useful results, the probability of traumatic consequences from a random error may be low as long as gross errors are likely to be detected manually before they result in serious harm. The primary concern is for accidents, errors, and omissions. Deliberate acts could occur, but there is little motivation for them, and those responsible for the system may well decide that security efforts will be more effective if they are concentrated on oversights and errors. If personal or proprietary data is involved, then data confidentiality is a concern. ADP availability will be a requirement for some of these applications.

2.3.2 Funds Disbursement, Accounting, Asset Management Systems

MAJOR OBJECTIVES: data integrity

EXAMPLES:

* Payroll
* Benefits payments
* Accounts payable
* Accounts receivable
* Inventory control

In these applications, deliberate and accidental acts are a major concern. While errors are the major cause of losses, these systems must also be protected against fraud, embezzlement, and theft. These systems frequently involve personal or other confidential data and thus data confidentiality may also be required. ADP availability objectives depend on the consequences of delayed action and on the effectiveness of manual fallback procedures [AACO 78], [AICPA 77], [PWCO 79].

2.3.3 General-Purpose Information Systems

MAJOR OBJECTIVES: data integrity and data confidentiality.

EXAMPLES:

* Generalized data management systems
* Centralized management information systems

These applications, which serve large and varied user populations, have the data integrity concerns of any of the above systems. Furthermore, the simultaneous use by different user populations makes data confidentiality more critical, and the accumulation of large quantities of data in the same system may make its misuse more attractive. The generality of these systems and their associated security requirements also make it more difficult to design effective security controls. Availability objectives come from the user groups that are most dependent on the system for their day-to-day functions [FIPSP 80B].

2.3.4 Automated Decisionmaking Systems

MAJOR OBJECTIVES: rigorous data integrity.

EXAMPLES:

* Fully automated funds disbursement and accounting systems
* Automated inventory reordering
* Automated scheduling for maintenance

In these systems manual review becomes more difficult, so that errors made by the automated system are less likely to be detected before they lead to serious harm. The degree of data confidentiality and processing availability required depends upon the sensitivity of the application's data and upon the consequences of delayed processing respectively [RCBCG 76A].

2.3.5 Real-Time Control Systems

MAJOR OBJECTIVES: ADP availability and data integrity.

EXAMPLES:

* Air traffic control
* NASA mission control
* Rapid transit system control
* Automated production control

These systems have all the problems of automated decisionmaking systems plus a rigorous requirement for constant availability and very rapid response times. The security controls discussed in this document are needed in these systems, but additional controls including automated fault detection, backup, and recovery along with redundant hardware support are also needed to meet all their security objectives.

2.3.6 Systems Affecting National Security or Well-Being

MAJOR OBJECTIVES: data integrity, confidentiality, and ADP availability.

EXAMPLES:

* Military command and control
* Management of multilevel classified information
* Integrated electronic funds transfer
* Nuclear material control and accountability

These systems must be protected against hostile acts by unauthorized persons who have considerable resources to devote to planning ways to compromise them. The approach to security discussed in these guidelines is not effective against this kind of threat. More effective techniques for formal development and verification of controls in both the operating system and the application, which are emerging from the research community, are necessary for improved protection in this area [BERST 79], [DODKS 78], [FEIER 79], [LINDT 76C], [MCCAE 79], [NEUMP 77B], [POPEG 78A], [POPEG 79B], [SCHIW 75], [WOODJ 77], [ARMY 78], [JEFFS 76], [NCEFT 77], [WULFW 74B].

2.4 Vulnerabilities

Information resident on a computer system is vulnerable to a wide variety of hazards. The selection of controls must be based not only on security objectives but also on all the vulnerabilities of the application. Some familiarity with these vulnerabilities is required in order to understand the problems being addressed by these guidelines. A few general examples are given below. The reader is referred to the appendix of [FIPSP 79B] for more extensive examples.

* Input errors—Data may be altered, disclosed, lost, or misidentified during input preparation and incorrect, misinterpreted, inconsistent, or unreasonable data may be accepted as valid for processing.

* Open system access—There may be no control over who can use the computer application.

* Poorly defined criteria for authorized access—The personnel in direct charge of the application may not know who should have access to various items of information.

* Unaudited access to data—If individuals can access data knowing that there will be no record of their actions, then they will feel they cannot be held accountable for their acts.

* Unprotected information—Data may not be protected from unauthorized access. This applies to on-line files and also to off-line files such as magnetic tapes. The latter are sometimes accessible simply by requesting that they be mounted.

* Dial-in access—Unauthorized persons may obtain access to the system—especially if remote, dial-in access is allowed.

* Program errors—Programs can contain many undetected errors, especially when they are written with poor programming practices or are not extensively tested. A program error may result in undesirable modification, disclosure, or destruction of sensitive information.

* Mistaken processing of data—Processing requests may update the wrong data; for example, if a tape is mounted at the wrong time.

* Operating system flaws—Design and implementation errors in operating systems may allow a user to gain control of the system. Once in control, the user can disable controls, erase audit trails, and access any information on the system.

* Subverting programs—Programs containing hidden subprograms that disable security controls can be submitted. These programs can copy information into secret or misidentified files for use at a later time, or perform unauthorized modifications of data.

* Spoofing—Actions can be taken to mislead system personnel or the system software into performing an operation that appears normal but actually results in unauthorized access.

3. BASIC CONTROLS

This section describes the basic controls that can be used to achieve security objectives. Selection of the controls that are applicable and necessary for a given application system depends both on its security objectives and on the environment in which the system operates. Guidance on how to make this selection in an orderly way will be given in sections 4 to 7.

Some controls discussed in this section may be implemented partially by the hardware, by the operating system, or by the facility management. Any applicable controls that are already available should be understood and used. Some controls that are often supported by an operating system are discussed in this section because the boundary between the operating system and the application system varies considerably and because it is ultimately the application system designer's responsibility to ensure that the combined effect of all controls is adequate.

Controls that are often expensive are identified as such in their descriptions; however, the actual cost will depend on many factors unique to each facility. Since many controls will also improve the manageability and the reliability of the application, much of their cost should be considered an integral part of the development and operation of an ADP application.

3.1 Data Validation

PROBLEMS ADDRESSED:

Invalid data leads to erroneous outputs and is usually the proximate cause of harmful human, social, and economic effects.

Invalid data can destroy the credibility of the system, demoralize those trying to use it, cause excessive system maintenance costs, and, in extreme cases, cause the system to be unavailable or unusable.

Data validation involves the examination of computerized data to determine if it is accurate, complete, consistent, unambiguous, and reasonable. This section gives an overview of the direct methods that can be used to evaluate data in order to identify likely errors. Since these direct evaluation methods are not able to find all errors, data integrity also depends on the correctness and integrity of all the activities by which the data is collected and processed. The controls of sections 3.2 to 3.6 as well as those in sections 6 and 7 must also be used in order to prevent subtle errors from being introduced into the data. Data validation is a very basic control, but it should only be expected to detect gross errors and it will not compensate for poor control over other aspects of the application system.

Data should be validated:

* during data collection and entry—prior to its use by the system; and
* continuously, as new data is generated or used during processing.

Errors should be detected and corrected as soon as possible in order to prevent the propagation of invalid data throughout the system and the potential contamination of the system data base.

Once invalid data has been identified, it must be corrected. The process of correcting errors is itself very prone to further errors, and data validation methods should be applied thoroughly during the correction activity.

Input verification, consisting of operational procedures to assure that source data is accurately transcribed to machine-readable form, is an important aspect of data validation. Input verification is discussed in greater detail in section 7.1 along with other manual information handling controls.

3.1.1 Consistency and Reasonableness Checks. Relationships between data elements that arise from the intended meaning of the data can be used to identify errors in the source data as well as errors in data preparation and processing. Often past records that are already in automated form can be used to predict probable values for current records. Any major deviation from expected values can then be flagged for further investigation. The appropriate checking to identify suspicious values must be defined by those who are familiar with the application and must be included in the application's design; it cannot be left up to the programmers.

3.1.2 Validation During Data Entry. Automated editing and validation should be used in both batch and on-line systems. In a batch processing system, validation routines may be run against the input data before it is processed. Alternately, validation can occur as each transaction is processed with transactions that contain errors being recorded on a file for correction at a later time. On-line systems can provide the data entry personnel with immediate validation information so detected errors can be corrected immediately. In either system, checks that must also use information from the data base may not be practical until just before processing.

Basic techniques for editing and validating data during data entry are listed below. More thorough discussions of techniques can be found in [MARTJ 73], [MAIRW 76], [JANCE 74], and [IIASA 77].

* Fields can be checked for:
  - legitimate characters (format checks),
  - proper sequences with respect to corresponding fields in preceding data records,
  - reasonableness,
  - consistency with other fields in the record,
  - values within specified ranges or limits,
  - valid items or codes,
  - the validity of a self-checking number contained in an extra digit(s) appended to the field, and
  - cryptographic check digits (i.e., data authentication information appended to a field which is a cryptographic function of all the data in the field).
* Records or transactions can be checked for:
  - completeness,
  - consistency of fields in the record with items in previous transactions or with other items in the data base, and
  - valid sequences of transactions.
* A group of records or transactions can be checked by maintaining:
  - control totals (e.g., batch, hash, amount, document, running),
  - record/transaction counts,
  - batch number checks, and
  - item/transaction balances.

3.1.3 Validation During Processing. Further checking of input data during processing and validation of data that is generated by the application system are essential for assuring data integrity. Many of these techniques are also used by auditors to audit data integrity. All of the methods described previously for validation of input data can be applied during processing. In addition, the following methods can also be used. More detailed discussions of these can be found in [MARTJ 73], [MAIRW 76], [JANCE 74], and [IIASA 77] referenced above.

* Integrated Test Facility (ITF). The ITF allows the performance of the application system to be monitored by incorporating dummy master records into the data base. Once these records are in place, test transactions which are included with live data can be processed against the dummy records during the normal processing cycle. Processing results of the test transactions can then be compared to predetermined results.

* System Control Audit Review Files (SCARF). Specifically designed tests—placed within the application's code—are performed on the processed data during normal processing. Some criteria are selected to extract desired records (e.g., processing exceptions, predetermined samples) and to write them onto a file which can be reviewed at a later time.

* Tracing provides the ability to follow (trace) specifically marked or tagged transactions as they are processed by the application system. It requires additional application codes and an extra field in the transaction for the tag. During processing, a record or trail is generated for the marked transaction which can be analyzed to determine if the processing was correct.

* Sampling techniques where portions of the data are inspected to determine their validity.

* Run number controls which account for each run of an application system against a data base. The run numbers kept in the data base and in the application program must agree.

* Reconciliation of control files. Important elements of the data base are identified and a file of transactions is maintained with a running total. Periodically, the old balance (taken from the last reconciliation) is added to the running total and must equal the new balance (taken from the data base).

* Off-line file scanning of direct access files during periods of their inactivity (e.g., at the end of the day) can be used to balance files.

* Rounding error controls to prevent fractions of cents from being stolen.

3.1.4 Data Element Dictionary/Directory. A data element dictionary/directory (DED/D) is a software tool associated with data base management systems which can be used to validate data elements in a data base. (In addition to its data validation function, it can also contain security authorization specifications to control access to data elements, and it can aid the computer auditing function.)

The DED/D can contain validation and redundancy-checking criteria in the form of formatted descriptive information defining both legal formats for data elements and expected relationships between data elements in the data base. More detailed descriptions of DED/D can be found in [LEONB 77], [ICST 77], and [FIPSP 80A].

3.1.5 Management Considerations. Data validation can be costly, and it may not catch the most subtle errors. Nevertheless, data validation is basic to good system design; it is not just a security control. Consequently, data validation costs should be included in the cost-benefit studies of the total system.

3.2 User Identity Verification

PROBLEMS ADDRESSED:

Identity verification is necessary if access to the system's data and resources is to be controlled.

Users cannot be held accountable for their actions unless they are positively identified.

User identity verification usually involves a two-step process:

* Identification which occurs when the user provides an identifier—the name by which the user is known to the system. The user's identifier is unique, is unlikely to change, and need not be kept secret. It is used during processing for authorization controls, for variance detection, and for other purposes such as accounting and billing.

* Verification which occurs when the individual passes some further test which "proves" that the user is actually the person associated with the identifier. This is also called user authentication.

3.2.1 Requirements for Identity Verification. Requirements for identity verification depend on the nature of the interface to the application system. For batch applications the requirement may be simply for visual identification and manual logging when the job is submitted. For applications with on-line users, it is usually necessary to require automated identity verification. Highly privileged actions such as modification of the software and changes to authorization tables usually should be allowed only from a controlled area where redundant visual and automated identity verification can be required.

Users should be uniquely identified so that they can be held responsible for their subsequent activities—it is usually not enough just to verify that the user is one of the authorized users. The only cases where it may not be necessary to have distinct verification test for each individual occur when:

* All approved users of a given interface are allowed to perform any of a (possibly restricted) set of actions (i.e., no authorization controls from sec. 3.3 are needed).

* There is no need to keep records of who performed what actions (i.e., no journalling and variance detection controls from secs. 3.4 and 3.5 are needed).

The second condition is seldom fulfilled since it implies that any authorized user can misuse the system anonomously and with very little risk of detection or that recovery from system failure does not require knowledge of which users performed what prior actions. Nevertheless, in situations where employees have to share a terminal to make brief queries, repeated identity verification could be burdensome. If there are physical and visual controls to ensure that unauthorized users cannot get to one of the terminals and if a careful analysis shows that journalling and variance detection on the queries is not cost effective, then unique identification of each user may not be necessary.

3.2.2 Basic Techniques. There are three basic techniques by which an individual can "prove" identity. The individual can present for verification:

* Information KNOWN
  - password
  - numerical combinations
  - other information known only to user

* Objects POSSESSED
  - magnetically encoded card
  - key for a lock

* PHYSIOLOGICAL, BEHAVORIAL, or MORPHOLOGICAL attributes
  - signature
  - fingerprints
  - voiceprints
  - appearance and facial characteristics

Of all three basic techniques, passwords (something KNOWN) are the most commonly used automated method of verifying a user's identity. If password generation and password distribution are effectively administered and if passwords are adequately protected by users and by the system, then passwords can be used to verify identity with a degree of reliability that is adequate for many applications. A more thorough discussion of password techniques is contained in [WOODH 77] and [HOFFL 77].

Of the three techniques, the last two automated approaches are usually more expensive than passwords but may be somewhat more reliable. When necessary, they can be used in combination with passwords to obtain a degree of reliability commensurate with the risks of most applications. A detailed description and guide for evaluating automated identity verification techniques can be found in [FIPSP 77B], [HOFFL 77], and [CARRJ 77].

3.2.3 Management Considerations. None of the verification techniques mentioned above are completely reliable since:

* passwords can be lost, stolen, or revealed,

* cards and keys can be lost, stolen, or duplicated, and

* devices used to recognize physiological attributes will accept a small percentage of impostors and reject a small percentage of authorized users.

Although strictly enforced administrative controls and user cooperation can greatly enhance the reliability of verification techniques, high reliability can be costly and entail an inconvenience for users. Consequently, identity verification requirements should be tailored to the actual needs of each system interface.

3.3 Authorization

PROBLEMS ADDRESSED:

If several people are to use an application that maintains diverse information, then data confidentiality and data integrity will not be preserved unless each user is allowed access to only that information which is needed.

Once a user's identity has been verified, the application may still need to check each request for service to determine whether it is a legitimate request by that user. Some users may be authorized to perform some functions but not others and to have access to some data but not to other data [SALTJ 75], [DENND 79B], [REEDS 74], [BERGJ 72], [CARRJ 77].

An elementary level of authorization can be enforced if the application has different interfaces for different purposes. For example, if there are different interfaces for entering transactions, for generating reports, for updating data, and for modifying programs, then each user can easily be authorized to use some but not all of these interfaces.

More complex authorization systems will usually be needed in order to satisfy each user's actual needs without allowing unneeded access to the application's data, programs, or other resources. In general, the principle of "least privilege" should govern. The less a person using the system is allowed to do (consistent with the work the user is required to do), the safer will be the application's other users, and the individual's own data and resources.

3.3.1 Authorization Schemes. Authorization schemes will have to be tailored to the needs of the organization being served by the application. There are a wide variety of possible authorization schemes described in the literature on computer security [MARTJ 73], [HOFFL 77], [SALTJ 75], [FABBR 74], [FERNE 75A], [FERNE 75B], [GRAHG 72], [HOFFL 73], [CHAMD 75], [STONM 74], [REEDS 74], and [LINDT 76C]. In general, an authorization scheme is the means whereby management can control:

* WHO will have
* what MODES OF ACCESS to
* which OBJECTS and resources.

WHO may include not only the human users of the application but also terminals and modules within the application. In many cases access to some sensitive objects should only be allowed from certain terminals. It is also possible for an authorization system to control the interactions between modules internal to the application software. This can greatly enhance the integrity, reliability, and maintainability of the application software [LINDT 76C]; however, this use of authorization is not yet widespread.

The different MODES OF ACCESS typically include read, write, and execute access to data and programs. The allowable modes of access can be usefully extended to provide independent control over access to each function supported by the application or by its internal modules.

The OBJECTS which are candidates for authorization control include:

* Data objects (files, records, libraries, segments, etc.),
* Executable objects (commands, programs, etc.),
* Devices (terminals, printers, etc.),
* Storage media (tape reels, disk packs, etc.),
* Transactions,
* Control data within the application, and
* Named groups of any of the above elements.

The ability to create named groups of system elements is an important way to simplify the administration of an authorization scheme.

In some cases, the authorization decision may depend not only on WHO is requesting what MODE OF ACCESS to which OBJECT, but also on other easily testable conditions. The time of the day, the day of the week, previously detected security variances, or other concurrent activity might be used to affect the authorization decision. In some cases, not only the name of the object but also a current value stored in the object may be used to make the authorization decision. This is called data-dependent access control. Clearly, the more factors that are used in arriving at an authorization decision, the more complex will be the required mechanism.

An authorization scheme consists of two parts—a set of authorization data or "rules" and a mechanism that uses the authorization data to decide whether a given access request should be allowed or denied. The authorization data allows the authorization system to accommodate to change whereas the authorization mechanism is built in and is usually very difficult to change. A careful study may be needed to select an authorization scheme which can accommodate all likely requirements over the lifetime of the system and yet is not so complex that it is unreliable and hard to manage.

The authorization mechanism generally should be invoked when one named element refers to another (e.g., a user request for service, a transaction input, a call for execution, an application system call on the operating system). The mechanism should be invoked at the point where the controlling process resolves symbolic references.

3.3.2 Delegation of Authority. An authorization system should distinguish the right to interact with an object from the right to authorize others to interact with it. In general, the three authority levels:

* ability to interact,
* ability to authorize interactions (delegation), and
* ability to appoint those who may authorize interactions,

should be separately controlled. None should, except at management option, imply either of the others. This is analogous to the distinctly different authorities involved in entering a bank vault, guarding the vault (deciding who may enter), and appointing the guards.

Policies for delegating authority should be given special attention when designing an automated authorization system. Delegation policies that appear to be simple can have surprising secondary effects in an automated system. For example, consider an individual with authority to access an object and to delegate access rights to others. If the user is about to have access authorities rescinded, the consequences must be well understood. One consequence (depending upon design) might be that all users whose authority originates from that individual will lose their authority when the primary user's authority is removed. This may be acceptable in some cases, and it may be catastrophic in others. On the other hand, if the authorities originating from the primary user's authority are undisturbed and untraceable, this too can be catastrophic. The authorization mechanism should enable management to discern the ultimate effects of such removals or modifications of authority. This requires that the authorization data, including back-chained or derivative authorities and all grouping relationships, be displayable.

The concept of data "ownership" is often used in security applications. The "creator" or "custodian" of a data file is usually considered to be the "owner." Other people who use the data are given the right to read it or to update it under controlled circumstances. Access may be granted temporarily or permanently (i.e., a copy may be made and the recipient becomes the "owner" of the copy). Security provisions must take into account the desirability of implementing the concept of "ownership."

3.3.3 Protecting the Authorization Data. The authorization data must accurately reflect management's intentions with respect to authorized activities. The authorization data is the basis for controlling system activity and has great potential for disruption.

It should be simple for management to grant, modify, or rescind an authorization; and management should be able to display all the current authorizations. If these actions are complex or unwieldy, errors will be more likely, disruptions will be more frequent, and use of the authorization mechanism will be less likely. In the past, simple and flexible delegation and display capabilities have been perceived to enhance the manageability of the system quite apart from any security enhancement (if good management and security are indeed separable). Where they have not been offered, management has been reluctant to use the authorization mechanism.

The authorization mechanism, to the extent feasible, should be self-protective. Erroneous, anomalous, or inconsistent authorization data entries should be detected and reported as early as possible—hopefully, at the time of entry and at least at the time they are first used in normal authorization checking. Clearly, the authorization data must be carefully protected against any unauthorized modification.

3.3.4 Authorization Schemes for Confidential Data. Authorization is useful to protect data integrity as well as data confidentiality; however, when the primary concern is for data confidentiality, a relatively simple authorization scheme is possible [DENND 75], [DENND 76].

Access to confidential data can be controlled by tagging sensitive objects with confidentiality labels. These tags or confidentiality labels should be made up so that the same restrictions will be applied to all data with the same labels. A special case of the concept of confidentiality labels is used to protect national security information. The "Confidential," "Secret," and "Top Secret" classification levels as well as other special categories can be thought of as one form of confidentiality labeling. For other types of information, different labels can be used; for example, in a personnel system, the set of labels might include "medical," "psychiatric," "personnel evaluations," "disciplinary actions," and "payroll." Many different labels may be used and several labels may be associated with any one object.

The term "confidentiality label" is used to avoid confusing the general concept with more specific versions of the concept. The general concept is recommended, and it can be tailored to meet a variety of local needs.

Some labels will identify data that is very sensitive, and others will identify data that is less sensitive; for example, top secret data is more sensitive than secret data, and psychiatric data may be considered more sensitive than medical data. On the other hand, the labels need not form a strict hierarchy of sensitivity levels. The authorization scheme need not require a decision about whether medical data is more or less sensitive than personnel evaluations. The decision about who is allowed access to each type of information can be made without determining its relative sensitivities.

Users are authorized access to information by associating a set of labels with each user—thus establishing access privileges for the user. The simple access control rule is that users can access an object only if their set of labels includes *all* the labels that are associated with the information. In some cases it may be desirable to have access privilege labels for terminals and for internal program modules in order to control the information that they can access.

To implement hierarchic relationships between labels, a user who has a certain label should also have any labels that are strictly less sensitive; for example, if a user permitted to access psychiatric information is also permitted to access medical information, then the user's set of labels should include both "psychiatric" and "medical."

The primary advantage of confidentiality labels is that they provide an authorization scheme which is simple enough so that errors in enforcing the scheme are less likely. Delegation is especially simple since it is only necessary to establish confidentiality labels for each user. Furthermore, the same scheme can be applied throughout all associated manual activities. The primary weakness of confidentiality labels is that they are not readily usable to protect data integrity.

The number of distinct confidentiality labels that can be used will determine the flexibility of the authorization system. If there are only a few, centrally administered labels, then administration of the authorization system will be easy, but it will be limited in its flexibility to enforce the principle of least privilege. If there are a large number of labels, and especially if they can be created dynamically, then more flexible control over confidential information is possible.

Confidentiality labels should be carried along with information whenever it is moved from one location to another. (Within a software implementation, confidentiality labels need not be stored with the information as long as they can be determined reliably when needed.) The confidentiality labels should appear on all output, and the authorization scheme should be applied in all the manual information handling activities associated with the application.

3.3.5 Management Considerations. Since damage to the authorization mechanism or data can disable operations, some bypass mechanism or procedure may need to be available so that management has the option to continue operations in an unprotected mode. The designers of authorization mechanisms should not assume that management uniformly believes that system shutdown is preferable to interrupted or degraded protection. Any bypass mechanism is sensitive and dangerous and must be shown to be safe from unauthorized use.

It can be a traumatic experience to transform from a system with little or no authorization control to one that is heavily controlled. In some cases, management simply does not possess the required specific information to create the complete set of authorization data. Thus, a gradual transition is indicated. One approach is to include a capability such that initially, while all checking based on the growing body of authorization data is done normally, authorization "failures" do not cause denial of the requested interaction. Instead, a record of the failure is kept for subsequent analysis. In this way, management can accumulate the appropriate authorization data without disrupting normal operations. As confidence in the authorization data increases over time, the normal authorization failure processing can be used increasingly until the system has been fully converted.

3.4 Journalling

PROBLEMS ADDRESSED:

It is difficult to detect security violations unless there is a record of system events which can be analyzed.

Users are less likely to engage in malicious activities and to make errors in their day-to-day interaction with the system if they know that their activities may be recorded.

Recovery from system failures is aided by journalling appropriate events.

The recording of selected events as they occur within a system, referred to as journalling or logging, provides a basis for identifying and tracing events involved in the processing of data and in the use of computer resources

[BJORL 75], [CARRJ 77], [KRAUL 79], [IIASA 77], [MARTJ 73], [HOFFL 77]. Journals of external events in the operational environment can be maintained manually while journals used to record activities internal to the application system must be automated.

One or more automated journals may be employed; the system designers should structure the journals to permit flexibility while minimizing overhead.

Journalling can produce a large volume of data that must be analyzed if it is to be meaningful. The analysis is part of variance detection which can be performed either dynamically as the journal is created or after the fact through a post-processing analysis of the journal. In addition, the journals gather information that can be used to optimize the use of system resources, to reconstruct data bases after system failures, and to audit the system by following the history of a transaction through the system.

3.4.1 Contents of the Journal. From a security standpoint, the ideal journal would include a 100 percent recording of all events relating to data, software, and system resources. From a practical standpoint, such a journal may, in some systems, be out of the question as it could not meet the cost-benefit test[3]. The question of what should be included can only be answered in a general way; the journal must be designed for each individual system and be dependent upon the security objectives for the system and the environment in which the system operates.

In determining what should be journalled, it is important that security requirements be carefully considered and reduced to formal statements. The requirements should be stated in positive terms and include such items as:

* definitions of what kind of data is to be protected and how the system will recognize such data,
* the degree of accuracy that is necessary for various types of data, and
* the definition of who is authorized to access protected data and how the system would recognize an authorized user.

The contents of the journal should be the subject of a comprehensive study that uses the system's mandatory security specifications as a departure point. The results should define:

* the type of information to be recorded,
* the parameters to be applied,
* the analysis to be made,
* the reports to be produced, and
* the retention period of the recorded data.

Selecting the specific data to be captured and journalled is a managerial decision that should be made only after detailed analysis by both users and data processing personnel. Journalling and the associated processing of the journals can impose a performance burden. Consequently, cost must be considered in relationship to the benefits to be gained. The type of information typically included in journals is illustrated by the following:

* nature of the event
  - input and output of data (e.g., opening and closing of file),
  - updating, changing, or modifying data sets of files, and
  - system usage or attempted usage (e.g., logons and logoffs).
* identification of all involved elements
  - people,
  - devices,
  - software, and
  - data (indexes, directories, files, etc.).
* information about the event
  - time and date,
  - success or failure,
  - authorization status of involved elements,
  - transaction numbers,
  - system response,

[3] The most common argument against journalling all events is that it is too expensive. In some cases it may be worthwhile to journal all events and retain the journal for only a few days, provided the journal is promptly processed each day after acquisition.

- addresses of items updated,
- contents of items/records before and after creation, update, and deletion,
- programs used,
- results of compatibility and parameter checks, and
- procedural violations.

For automated journals, due consideration must be given to the problems of archiving extensive data for what may be prolonged periods, even years. The ability to easily off-load voluminous journal data, to condense it as much as possible, and to on-load easily the same data for inspection much later in time, perhaps on a different machine complex, is important. These capabilities may also be required for other reasons—especially for internal and external auditing activities.

3.4.2 The Journal as Legal Evidence. The journal can be a valuable indicator for alerting management when an employee is involved in fraud, embezzlement or other misuse of the system. In addition, it may be necessary to use information contained in the automated journals as evidence of the employee's illegal activities. Under the Federal Rules of Evidence, computer output (e.g., automated journals) is admissible in both civil and criminal proceedings if it is determined to be:

* a regularly kept, timely record
* of regularly conducted business activity
* whose preparation has been deemed "trustworthy" by the court.

The first two conditions are relatively easy to establish for a well-run enterprise; the third may present a problem if it cannot be shown that the records and the recordkeeping process itself are reasonably safe from accidental and unauthorized intentional interference. This showing is not only essential to the court's determination of admissibility, it is also an important defense against attacks upon the credibility of the output.

3.4.3 Management Considerations. In journalling, the level of detail monitored, the amount and types of data saved, the length of time the data must be saved, and the complexity of processing the data not only affect the security effectiveness of this tool, but also influence the cost and hardware requirements of the system. Care must be exercised when implementing this tool to prevent the system from being bogged down with journalling to the exclusion of useful work.

The journalling activity is vulnerable to accidental or deliberate events that can endanger the journalling process or the journal contents. Examples include:

* The journal contents may be partially or totally lost in a system crash caused accidentally (e.g., by a power failure to a volatile storage device) or deliberately (e.g., by someone trying to avoid detection).

* The journalling can be disabled or the journal destroyed by a system penetrator.

It is not always practical or possible to protect completely the journalling activity against such events. While it should be protected to the extent possible, management should be aware that the journal may not be available when it is most needed.

The journal produced by an application system will not log activities not performed by the application system; e.g., a file opened outside of the application system.

3.5 Variance Detection

PROBLEMS ADDRESSED:

Fraud, embezzlement and other misuse of the system and its data by authorized users of the system.

Undesirable modification or disclosure of information that remains hidden from management review.

The objective of variance detection is to allow management to detect and react to departures from established rules and procedures that it has determined may constitute hazards. Variance detection acts as a strong deterrent to authorized users abusing their privileges since they perceive the risk of detection to be unacceptably high [MARTJ 73], [CARRJ 77].

Variance detection is useful whenever it is not practical to prevent the undesirable activity by means of an authorization mechanism. In many cases there are tradeoffs between preventing and detecting an undesirable activity with the choice depending on the potential harm that could occur. In other cases there may be no way to determine in advance whether an action should be prevented.

Some suspicious activity can only be identified when the journal discloses a highly unusual pattern of activity. In such situations, the analysis may involve determining what information an individual has accessed over some period of time and ascertaining what the individual could be doing with that information. The complexity of the analysis may preclude prevention even though the activities can later be identified as suspicious. An example would be an employee who obtains a complete listing of sensitive information by piecing together small portions that have been gathered in repeated requests. Assuming that the employee is not authorized to obtain a complete listing, only an analysis of the journal files is likely to uncover the variance.

The mechanisms required to support variance detection are related to mechanisms needed for several other purposes. Recovery, accounting, load-balancing, tuning, and the identification of recurring user difficulties all require some of the same capabilities. The design may be such that one multipurpose mechanism can accomplish all or most of these ends.

3.5.1 Management Inspection of Event Journals. To facilitate management inspection of manual journals, it is important that they be:

* suitably formatted,
* legible,
* complete,
* timely, and
* correct.

For management inspection of automated journals, a post-processing capability should exist that offers both interactive query and report generation functions. This capability, often referred to as static monitoring, requires analytical and retrieval programs that perform functions such as listing:

* all or selected activity initiated from a given terminal or by a given employee,
* transactions of a predetermined type,
* all events falling outside predetermined parameters,
* attempted accesses to restricted files,
* excess resource usage,
* changes made to access privileges of users,
* program detectable software malfunctions,
* abnormal terminations of job runs, and
* requests for unauthorized use or modification of a program, file, or sensitive system data.

Post-processing analysis can perform complex correlations using the journalled information and the logs of system activity produced by the operating system. Patterns often can be determined that direct attention to a particular terminals, user, time of day, etc., and thus assist in ferreting out an error-prone or malicious user.

Reports and other output must be as concise as possible, and specifically pinpoint any unusual actions that occur. Long reports containing large amounts of data may actually decrease detection of security violations as they are tedious and difficult to use. While concise information is desirable, details should be available through use of the journals and retrieval programs.

Management must be able to specify what reports are to be generated periodically. In addition, special report generation functions or an interactive query capability to extract information from the journals are useful. These retrieval programs should support complex Boolean and arithmetic operations upon data elements, numeric values, and statistical data derived from the journals. If management does not have this processing flexibility and power, the likelihood is that the journal will not be inspected regularly; users will realize this; and the deterrent value will be lost.

3.5.2 External Variance Detection Methods. In addition to inspection of the journals, several other approaches can be used to detect variances that occur in the manual procedures and practices associated with the application system:

* observations (visual, electronic, photographic),
* employee interviews and questionnaires,
* undercover operations,
* surprise visits, and
* scrutiny of employee reports for anomalies.

3.5.3 Response to Variances. There must be clearly defined responsibilities to investigate security variances once they have been detected. Since responsibility for security may be divided among many organizations, there is danger that no one will be accountable for security variances when the source of a problem has not been clearly identified. In order to avoid involving senior management unnecessarily, the responsibility for investigating such security variances should be assigned to appropriate organizations. It is reasonable to have the responsibility for investigating security variances depend on the general type of the failure. For example, if a data base becomes garbled with errors, preliminary responsibility may be assigned to the user organization that generated the data unless it is shown that the data integrity was destroyed by errors in programs, operations, data entry, or other functions that are not the responsibility of the user organization. Data confidentiality variances may be assigned to the person who has responsibility for the system of records under the Privacy Act, and ADP availability failures to the ADP operations organization. Responsibility for identifying holes in the overall assignment of responsibilities will naturally fall to the organizations that assume the responsibility for investigating variances.

While direct responsibility for each aspect of security should be assigned to only one person or organization, others should have a clearly defined responsibility to report security variances that they can detect. For example, while source data accuracy is not the primary responsibility of data base administrators, they should be required to report the existence of inaccurate or incomplete records. Similarly, while computer operators may not be responsible for preventing unauthorized access to the computer room, they should be required to report suspicious activity.

3.5.4 Dynamic Monitoring. Some variances can and should be detected in real time so the response can be immediate. Dynamic monitoring is usually integrated with the journalling activity so that suspicious activity is reported immediately.

A fair amount of logic may have to be built into the application system software if suspicious events are to be distinguished from normal processing activities. In general, certain events can be designated as "trigger-events" which cause the monitoring routines to be invoked. The monitoring routines can then examine other factors to determine if the event should be reported or responded to. Flexibility should be provided to permit monitoring of selected events such as all transactions meeting certain criteria, activity initiated from any given terminal, and all or selected transactions initiated by any given user. In addition, monitoring routines can be activated periodically to try to identify unusual values or patterns of activity.

It should be easy for management to establish and change the "trigger-events." To reduce the overhead that is caused by excessive monitoring, only a limited number of trigger events may be used at any one time. The deterrence effect is still substantial as long as management varies the set of trigger events in some random way.

A number of optional reactions to suspected variances can be used; for example:

* real-time alerts to management by the application system software such as a warning bell and a message to a designated console,

* suspension or termination of an offending process with a message to the user that will not arouse concern while the variance is investigated,

* automatic invocation of special monitoring such as complete journalling of all activities and interactive traffic of the offending process, and

* real-time display at a designated console of the full interactive traffic of an offending terminal.

There is some danger that the monitoring software could be used to violate security, and it must be protected from all unauthorized use. Controls must be established to ensure that the monitoring activity is conducted only from secure terminals under the direct supervision of the organizational elements responsible for the monitoring.

3.5.5 Management Considerations. Variance detection is not foolproof, and management should not rely on it as the only safeguard. Nevertheless, variance detection can be very useful to encourage a general awareness of security and to discourage dishonest behavior.

Monitoring of system use can also be employed to identify changes needed to improve efficiency of the work flow in and around the system. This utilization can result indirectly in improved security of operations. If a system user is advised that management has detected a pattern of frequent errors in the user's conduct of certain activities and is offering help, there will be increased awareness on the part of the user that such activities are being reviewed. In this way, monitoring can be productively employed without the necessity of justifying it on the basis of detecting or inhibiting dishonesty on the part of the users.

Clearly, management must be very careful about the way monitoring activities are perceived by the employees. It must be perceived as necessary and even as helpful. If an event is investigated as suspicious and no evidence of wrongdoing is discovered, then the employees must be convinced that there will be absolutely no lasting effects. No-fault variances caused by employees should not be counted nor kept in records associated with them. In most cases, the employees should not even feel a need to avoid repeating the acts that were identified as a possible variance.

If there is even one instance where monitoring activities are perceived by the employees as being used unjustly to harm an employee, the effect on employee morale can be devastating. Unfair usage of monitoring information can even lead to vindictive acts which destroy everything that was being protected by the entire security program.

3.6 Encryption

PROBLEMS ADDRESSED:

Disclosure and undetected modification of information during its transmission through a potentially hostile environment, including intermediate nodes in a network.

Disclosure of information when tapes or other storage media are lost or stolen.

Encryption [FEISH 73], [KAHND 67], [POPEG 79A], [SHANC 49], [CARRJ 77], [HOFFL 77] transforms data into an unintelligible form, typically by the use of an algorithm and a key. Encrypted data can be decrypted (return to its original form) only by using a companion algorithm and either the same key in the case of a "secret" key system or a companion key in the case of a "public" key system [RIVER 78]. A standard "secret" key algorithm for encrypting and decrypting computer data is specified in [FIPSP 77A]. The key is a very large "secret" number that effectively prevents decryption of the data by anyone who does not know the specific key.

Encryption is useful to protect computer data while it is being transferred between people or devices authorized to have the data and while it is inactive in storage. Encryption protects data from unauthorized disclosure because anyone who does not know the correct key is unable to derive intelligible meaning from the encrypted data. Encryption allows even slight modification of encrypted data to be easily detected because any modification of the encrypted data will result in gross and unpredictable distortions of the data in its decrypted form. Encryption does not protect the data from destruction—in fact, it increases the chances that the data may be destroyed or unavailable because if the key is lost, the data encrypted with that key is lost forever!

The applications that are most likely to need encryption are those that transmit highly confidential data across communication lines. Applications that transmit financial transactions or other critical data may also need encryption if someone is likely to derive enough benefit from modifying the data during transmission to compensate for the risk and the cost of the effort. Encryption of data in storage is an alternative that may be more cost effective than other storage security controls—especially when appropriate support for encryption is readily available.

3.6.1 Encrypting Computer Communications. Encryption is the primary control to protect data from the hazards of communication systems. It can be used to forestall the consequences of electronic eavesdropping, intentional modification, misrouting, introduction of spurious messages, spoofing (falsifying the identity of the transmitter), playback of old messages, and for the authentication of senders and receivers and the protection of acknowledgements that the message was received or not received correctly. The primary problems when using encryption for this purpose are to ensure that the encryption does not interfere with communication protocols and to guarantee that the correct key is available to all authorized recipients of a message—and to no one else. For details, see [COLEG 78], [BRAND 78].

3.6.2 Encrypting Off-Line Storage. Encryption of data on tapes and other off-line storage media is an effective way to protect the confidentiality of the data without having to maintain rigorous physical control over the data storage media. Again, the primary problem is to ensure that the correct key will be available to decrypt the data when it is needed—and that it will not be known to anyone else. In general, the keys should not be stored on the computer because then they would be subjected to all the hazards applicable to computer data.

Encryption protects the data if the physical storage media are lost or stolen; however, it may not protect the data from unauthorized use through the computer system. Since the data must be decrypted whenever it is to be used, the encryption will not protect the data from anyone who can masquerade as a legitimate user of the data.

3.6.3 Encrypting On-Line Files. Since encrypted data cannot be processed until it has been decrypted, encryption of frequently used data will not be very efficient unless the encryption is supported by special features in hardware and in the operating system.

For encryption of on-line data to be useful, the keys may have to be stored on-line. Even if the keys themselves are always encrypted except when they are within the encryption device, it is still necessary to verify the identity of anyone requesting a key, to determine whether the key has been authorized, to journal the usage of a key, and to detect any variance in the usage of a key. In short, the primary controls discussed previously will still be necessary to protect the keys; however, encryption can reduce the scope of the protection problem because it is no longer necessary to ensure rigorous control over those areas of computer storage that contain only encrypted data.

3.6.4 Management Considerations. The cost of hardware that implements the encryption algorithm can be expected to decrease rapidly throughout the early 1980's. As more experience is gained in the use of encryption, the costs of planning for its use and of managing the distribution of keys will also decrease.

Before hardware and software implementations of the standard data encryption algorithm can be procured for Federal use, manufacturers must obtain a certificate of validation for their product from NBS.

If encryption is to be used at all, it should be used properly. A poorly planned use of encryption, or a poor system for managing keys will give a false sense of security.

4. SELECTION OF CONTROLS THROUGHOUT THE SYSTEM LIFE CYCLE

The selection of appropriate security controls depends not only on the sensitivity of the data and the degree of harm that could result from improper actions, but also on the vulnerabilities that arise in the environment where the application operates. Thus far, no standard methodology for selecting controls has been proposed for implementing applications in an ADP environment. However, a good discussion of the required considerations for selecting appropriate security measures is found in [ORCEM 78].

While the selection of controls is still an art based on experience and good judgment, it is possible to make this selection if it is not overly constrained by poor planning for security. When compared with the level of security that is often found in computer applications, dramatic improvement can be achieved as long as security controls are selected by experienced personnel who:

1. are familiar with the vulnerabilities of the entire application system—including both automated and manual procedures,
2. can take a system-wide view of the application and have the freedom to locate controls at the most appropriate points, and
3. understand common application system control objectives and techniques [IIASA 77], [MAIRW 76], [BIGGC 80], [AACO 78], [AICPA 77], [EDPAF 77], [SGCCA 75B].

These guidelines will be of direct help in achieving conditions (2) and (3). Each agency has to rely on its own personnel to fulfill the first condition and to select the appropriate controls. The remainder of this document describes how condition (2) can be achieved as long as security is considered at each step of the application system's life cycle. While it is common practice for system developers to think of system functionality first and to delay security concerns until later, many opportunities to include effective controls are lost if security is not considered early.

Basic guidance to fulfill the third condition is contained in section 3. Additional guidance about enforcing security controls during development and operations is contained in sections 6 and 7.

4.1 The Application System Life Cycle

The life cycle of a computer application consists of three identifiable phases—initiation, development, and operation [FIPSP 79A], [FIPSP 76A], [FIFED 77]. After some period of operation, the system will have to undergo an expansion or revision, and the life cycle is then repeated. It is desirable for the life cycle of the application to be independent of the life cycles of the computer hardware, of the operating system, of the computer facility, and of associated applications.

INITIATION PHASE. The initiation phase establishes the objectives and general requirements of the computer application. System planners consider alternative approaches for a target system. Based upon feasibility studies and cost-benefit analyses of the potential solutions, a decision to proceed with the development of a specific system is reached. If security is not taken into account during this initial planning, feasibility studies and cost-benefit analyses may be seriously distorted.

DEVELOPMENT PHASE. The development phase consists of four stages—definition, design, programming, and testing. While these are logically independent stages, in practice they may overlap substantially. Most decisions about which security controls to include must be made in an orderly way during these stages. In addition, there are software development controls that must be applied during these stages to ensure that the desired level of security is achieved.

OPERATIONAL PHASE. The operational phase begins once the system has been accepted by its intended users, and they become dependent on it to fulfill their organization's mission and responsibilities. Operational security controls must now be enforced during day-to-day operations.

4.2 Improving Security for Existing Applications

With some minor modifications, the life cycle approach to security applies to computer applications that already exist. When the system is already in its operational phase, the following three steps are recommended:

RISK ANALYSIS. It is best to begin with a thorough understanding of the risks and exposures in the present system. This should be accomplished by a risk analysis following the methodology of [FIPSP 79B]. The risk analysis will indicate how urgent and how extensive the revision must be.

IMPROVE OPERATIONAL CONTROLS. The risk analysis will usually uncover some vulnerabilities that can be corrected by improving the operational controls described in section 7. The improved operational controls will be necessary even after the system is revised to improve security, and they will usually reduce the risk significantly while more thorough improvements are planned and implemented.

REVISE THE APPLICATION SYSTEM FOLLOWING THE LIFE CYCLE APPROACH. Revisions of the computer application should pass through the phases of initiation, development, and operation of the revised system. The guidelines in each of the following sections should be used for the revision—in an abbreviated and less formal way when the revision is minor.

When possible, it is best to integrate security improvements with revisions that may be needed for other purposes to minimize making repeated small modifications which are likely to make a computer application more complex and less reliable. This applies both to the software and to manual procedures. Even though it may be desirable to implement the revision as a set of small evolutionary changes, the changes should be planned in a coherent way to prevent the degeneration of the system.

4.3 Auditor Involvement in the System Life Cycle

The need for effective internal auditing [USGAO 72], [RCBCG 79], [USGAO 74] in Federal agencies has been recognized by the Congress in a number of laws, particularly the Budget and Accounting Procedures Act of 1950 and the Inspector General Act of 1978. An internal audit is an independent appraisal activity within an organization for the review of operations as a service to management. It is a managerial control which functions by measuring and evaluating the effectiveness of other controls. The responsibility of the internal auditor includes verification and evaluation of controls, standards, and data processing results. One aspect of an internal audit, which is particularly relevant to the security of a computer application, is a computer security audit activity for the purpose of evaluating controls to ensure:

1. the accuracy and reliability of the data maintained on or generated by an ADP system;
2. the appropriate protection of the organization's information assets (including hardware, software, and data) from all significant anticipated threats or hazards; and,
3. the operational reliability and performance assurance of all components of an ADP system.

With the growing dependence of organizations on technically complex ADP systems, it has become increasingly clear that in order to perform an internal audit and, in particular, a computer security audit, the auditor has had to modify, extend, and supplement traditional audit approaches to manual systems and to broaden the scope of the audit to include review during all life cycle phases of computer applications. As evidence of this, GAO has recently published three supplemental standards for auditing computer-based systems [USGAO 79] which require that:

"1. The auditor shall actively participate in reviewing the design and development of new data processing systems or applications, and significant modifications thereto, as a normal part of the audit function.

2. The auditor shall review general controls in data processing systems to determine that (A) controls have been designed according to management direction and legal requirements, and (B) such controls are operating effectively to provide reliability of, and security over, the data being processed.

3. The auditor shall review application controls of installed data processing applications to assess their reliability in processing data in a timely, accurate, and complete manner."

These standards reflect the current belief of the audit community [RUTHZ 77], [RUTHZ 78], [IIASA 77], [RCBCG 79] that adequate and effective security of computer applications can be achieved only by audit review and evaluation throughout the development and implementation of an application. This recommended practice ensures that:

* adequate controls are built-in to the application;
* the capability to adequately track events during systems operation is provided to facilitate future audit reviews; and
* accepted accounting principles are complied with for financial applications.

5. PLANNING FOR SECURITY DURING THE INITIATION PHASE

Decisions made during the initiation phase [FIPSP 79A], [BIGGC 80] will determine:

* The role of the ADP system and the degree to which the organization will rely on it to fulfill its mission.
* The possibilities for effective manual review of results from the ADP system.
* The nature and sensitivity of the data to be stored in automated form.

Security concerns must not be neglected as these decisions are being made.

In general, the relationship between the computer application and associated manual activities should be planned so that errors in one activity can be detected by the other. The computer application should be able to detect erroneous or fraudulent manual activity; and, when possible, manual activity should be planned so that critical output from the automated system is reviewed before final action is approved.

The initiation phase for revising an existing application is different from the initiation phase for a new application. When controls must be added to an existing system, cost constraints are different and the planners will find that they have fewer viable options open to them. Some controls may be reasonably simple additions to the existing system. Other controls will require major changes in the software design or in the external environment of the application system; and implementation of these controls will be costly unless a major revision of the system is needed in any case.

Before the planners decide to limit the scope of a revision, they must be sure that the easy controls are the controls that are needed. Money spent on simple enhancements might be wasted if the most serious vulnerabilities cannot be corrected without a more thorough overhaul of the system. The security feasibility questions of section 5.1 should all be asked as part of the risk analysis for an existing application. Some difficult decisions will have to be faced if the answer to any of these questions is not a clear "yes." Negative answers to these questions usually mean that a major revision of the computer application should be planned; however, in some cases it may be better to plan a sequence of revisions that will culminate in a system with a coherent and effective set of controls.

For new systems, there is a serious tendency to neglect security and integrity concerns until the nature of the system is clearly defined. Unfortunately, this can lead to systems whose basic characteristics are inconsistent with appropriate security objectives. The following two sections will help to identify security problems that are inherent in a proposed system. These sections should be used as part of the feasibility studies and the cost-benefit analyses, respectively.

5.1 Security Feasibility

PROBLEMS ADDRESSED:

Unrealistic plans for a computer application may result in security problems for which there is no cost-effective solution.

System designers and implementers will not perform miracles—it cannot be taken for granted that they will be able to build inadequate security controls. The five questions below cover the major concerns about security feasibility. If satisfactory answers are not available for these questions, system development should not proceed according to the proposed plan.

5.1.1 Source Data Accuracy. Will the data supplied to the ADP system be accurate and complete enough to support its intended uses without harmful side effects [RCBCG 78C]?

Source data will not be perfect, and additional inaccuracies may be introduced before the data comes under the control of the ADP system. Even if the overall error rate is very low, serious harm can result from a few errors if they have been deliberately introduced or if they involve critical data.

If the proposed system will use the data in the same way as an existing (manual or automated) system, then it should be relatively easy to evaluate the potential harm from inaccurate or incomplete data. If the proposed system uses new data sources or uses present data for new purposes, then a more serious study is needed.

In an automated system, the lack of human review of the data leads to some increased dangers; however, the ADP system can also contribute to improved data quality to the extent that automated data validation can be used to detect and reject inaccurate and incomplete data. The effectiveness of automated data validation techniques in identifying harmful inaccuracies is very application dependent and must be evaluated for each application system.

5.1.2 User Identity Vertification. Can users of the system be adequately identified and authenticated so they can be held accountable for their actions [FIPSP 77B]?

If former systems did not require users to identify themselves, then a major change in organizational procedures may be necessary. Furthermore, can outsiders and other unauthorized individuals be prevented from using the system? Rigorous control over who uses the system is especially important if the system is to allow access from dial-in telephone lines.

5.1.3 Restricted Interfaces. Are the user interfaces to the system sufficiently restricted so that adequate security is feasible [HOFFL 77], [CARRJ 77]?

If there will be a large number of authorized users, then the proposed system should restrict most users to limited forms of access to the system. Data accuracy will be very difficult to maintain if many different people can update a master data file. Even query rights to a data base should be examined for consistency with privacy and security requirements before they are planned for widespread use—especially by persons from outside the immediate organization served by the application. Since current operating systems do not prevent a maliciously written program from bypassing all available security controls, high-risk applications should be planned so that a very limited number of people will be allowed to submit programs to the computer facility that is running the sensitive application.

5.1.4 Separation of Duties. Do the boundaries between ADP and related manual activities provide maximum separation of duties and independent review [IBMCO 76]?

Errors are more likely to be detected and frauds deterred when two or more independent people are involved in all important actions. Security is easier if the department supported by the ADP system maintains controls over ADP and related manual activities. In most cases, data collection, initiation of transactions, authorization of expenditures, and physical control of assets can be maintained by the user department[4].

5.1.5 Facility Security. Is the proposed processing facility adequately secure [FIPSP 74], [RCBCG 76C]?

Selection of the computer facility need not be done before the design stage; however, if there are any assumptions or constraints on the possible facilities, they should be reviewed for consistency with security. Are physical security measures, operational controls, personnel and administrative procedures, emergency plans, and backup and recovery procedures at the proposed facility adequate for this application? Are the availability requirements of this application consistent with the performance of the proposed facility? For high-risk applications, is it acceptable to run the application concurrently with the other applications at the facility?

5.2 Initial Assessment of Risks

PROBLEMS ADDRESSED:

If security risks are not considered, cost-benefit analyses may favor a system plan with unnecessary exposures to traumatic failures.

A risk analysis [FIPSP 79B], [LOSOT 79], [WONGK 77] determines an annual loss expectancy which should be taken into account in the cost-benefit analysis. Furthermore, a risk analysis is a useful first step toward estimating the maximum tolerable cost of appropriate controls.

[4] The user department will need some redundant manual records if its controls are to be effective. These redundant records can also be used for backup and for auditing; however, paper copies should be kept only if it will be practical to use them when needed or when such retention is required by law or regulation.

When planning revisions for an existing application, a risk analysis for the proposed revision can be obtained by adjusting the risk analysis for the current system. The remainder of this section applies primarily to new applications or to applications undergoing major revisions.

A thorough risk analysis, including safeguard selection, should be performed at the beginning of the design phase to assure that appropriate cost-effective security controls are integral to the system's design. The initial assessment of risk should only estimate the damage that could result from major failures that might occur in spite of controls. With appropriate controls, such failures should be rare; however, some possibility of failure will remain. When possible, plans for the computer application should be revised to minimize the potential impact of such failures.

A risk analysis involves an estimate of the impact of a security failure and an estimate of its frequency of occurrence. The product of these two factors gives an annual loss expectancy. This annual loss expectancy from major failures, derived from an initial risk analysis, should be included when considering the cost-benefit tradeoffs between alternative systems.

5.2.1 Impact of Major Failures. For the purposes of this initial assessment, the impact of a security failure can be described in any convenient terms—dollar value of loss, extent of inconvenience or hardship to individuals, lives lost, or degree of disruption to the national economy or national security. As long as major impacts are not being overlooked or minimized, it is possible for potential impacts to be identified quickly.

The impact of at least the following failures should be assessed for each major body of information that is to be processed by the proposed system.

* *Inaccurate Data.* Data (or programs) become corrupted with errors, but the system continues to function while producing erroneous outputs. Estimate the potential impact of the erroneous actions that might result assuming only that the output of the ADP system is not so obviously out of line with reality that the errors would be noticed. Consider both the impact of a few very serious errors and the cumulative effect of many small errors.

* *Falsified Data.* An individual manages to falsify information in order to gain some advantage. The falsifications are limited only by the fact that they are subtle enough so they are not detected manually. Estimate the total impact that could occur over an extended period of time. Example: In a funds disbursing system, the potential impact might be limited to some percentage of the funds being handled. This limit assumes that losses over that percentage would be noticed based on budget projections and that such losses would eventually be corrected.

* *Disclosed Data.* Sensitive data in the system becomes available publicly or to certain individuals. The unauthorized disclosure of the data is not necessarily discovered.

* *Lost Data.* Data (or programs) used by the application system are destroyed or corrupted. Backup versions are nonexistent or not usable, and the data must be reconstructed manually. Estimate the impact of losing the data. If manual reconstruction is obviously not feasible and if backup in depth is anticipated, then estimate the impact of the inaccuracies and the temporary unavailability that would result while recovering from an old backup copy on the assumption that all recent backup copies had been destroyed.

* *Unavailable Data or Services.* The computer hardware or related equipment in the computer facility is disabled and the system is not available until it can be brought up on another facility.

5.2.2 Frequency of Major Failures. An estimate of the impact of a major security failure is not meaningful without some estimate of how frequently it might occur. Unfortunately, during the initial planning for a system, it is not possible to estimate the frequency of a major failure by evaluating the effectiveness of the controls because those controls have not yet been designed. Nevertheless, it is possible to make a very rough estimate of the likelihood of a major failure. If the proposed system is generally comparable to other computer applications, then a major security failure of the sort described above can be estimated to occur about once in a hundred years. This simple estimate is based on the following line of reasoning:

> Available security controls (if they are managed properly) can prevent *major* security failures from recurring as frequently as once every 10 years. On the other hand, any ADP system has several vulnerabilities against which there is little defense; most systems can be manipulated by any one of several individuals who are in a position of trust—programmers, those responsible for security, the computer operators, and others. There are enough instances of major security failures in computer applications so that an expected frequency of once in a thousand years is very optimistic. The estimate of once in a hundred years is only intended to be accurate to within an order of magnitude.

This estimate for the frequency of a major security failure can be used to give some perspective on the estimates about the impact of a failure and to compare the risks of an automated system with those of a manual one. Since the estimates of frequency of occurrence used during the initiation phase will be the same for all automated systems, only the estimates of the impact of a major failure will be useful to differentiate between alternative plans for automated systems.

6. BUILDING IN SECURITY DURING THE DEVELOPMENT PHASE

Once a decision has been reached to proceed with the development of a proposed application, the development passes through the stages of definition, design, programming, and testing [FIPSP 76A]. This section describes the security concerns that must be dealt with at each stage.

In addition to the specific security concerns described in this section, the security and integrity of an application system are strongly affected by the overall quality of the development effort [FIPSP 76A], [FIFED 77], [BIGGC 80], [MCCAJ 77A], [MCCAJ 77B], [MCCAJ 77C], [BOEHB 73B], [MILLH 77]. Effective management of the development activities is one way to improve security while at the same time reducing costs. If the software development effort is poorly organized and depends mostly on debugging and testing for its quality control, then software controls are not likely to be effective, and development costs are likely to be excessive. The following guidance specific to security should be used in conjunction with the more general guidance on managing software development in [FIFED 77].

6.1 Definition of Security Requirements

PROBLEMS ADDRESSED:

Many security problems can be traced to a poorly thought out security plan or to an inadequate definition of what the software was supposed to do.

The decisions about what security provisions are needed and to what extent they are to be enforced by software must be made and documented before the software development can proceed. The documentation of these security requirements constitutes the security specification called for in OMB Circular A-71, Transmittal Memorandum number 1, paragraph 4.c(1). This security specification may be incorporated into the functional requirements document and the data requirements document [FIPSP 76A], or it may be an independent document.

Programmers should not be expected to identify the security concerns that arise from the user's application. Security requirements should be defined by personnel from the organizations that are to be served by the application system as well as by computer professionals who have experience developing similar systems. Internal auditors, who are responsible for evaluating the adequacy of controls, should be called upon to review the security requirements. The proper completion of security requirements definition is more important and more difficult than is usually anticipated.

The term "requirements" can be used at many different levels. There are very general requirements for any government system to protect the rights of individual citizens and to manage government resources prudently. More detailed requirements come from laws, policy directives, and standards. The requirements defined at the beginning of the development phase are even more detailed and include everything that the users and other responsible parties want to require from the software system.

These requirements should be expressed in a way that allows the software designers to choose the best way of implementing them. Selection of software modules, algorithms, and data structures should be left to computer professionals who are competent to make those decisions; however, the requirements should define all details about the intended application which are not in the primary domain of knowledge of computer professionals. For example, in a financial accounting system the cross-checking of records that is part of standard accounting practice should be a requirement since it is more in the domain of knowledge of accountants than of computer professionals. Similarly, journalling requirements should specify the type of information discussed in section 3.4.1 rather than the algorithms and data structures for implementing the journals.

Some security controls can be enforced either by software or by physical and administrative procedures. For example, data integrity can often be checked either by human review or by automated bounds and consistency checks. When controls can be implemented in software, there are two reasons why this is preferable:

* Once controls have been automated, their continuing cost is usually lower than if they are enforced manually.
* Automated controls will be more consistently applied.

Unfortunately, the advantages of software controls can be lost if they are not implemented carefully. Since software does not adapt to unforeseen circumstances, there is considerable danger if software controls are incomplete or if they can be bypassed with unacceptable probability. Furthermore, if they are added after much of the application software is already designed or operational, the difficulty of modifying software can make their implementation quite costly—and can also lead to compromises that affect their effectiveness and reliability.

The following steps can be used to define security requirements and arrive at a detailed security specification:

6.1.1 Application System Interfaces. Identify each job function which is related to the application system. Consider each job function which relates to the application in a different way as defining a different interface to the system. Define the nature of the interaction between each job function and the system. Also identify and define any interfaces to other automated systems. Include all job functions (or other automated systems) that are to be supported by this system—even if the people performing those jobs only receive reports from it. Also include all job functions (or other automated systems) that supply information to the application system or that support its operation. Be sure to include critical job functions such as:

* Source data collection,
* Input preparation,
* Data entry,
* Output dissemination,
* Data base administration,
* System security planning and control,
* Internal audit,
* Application program maintenance,
* Archival or backup data storage,
* Computer operations, and
* System programming.

6.1.2 Responsibilities Associated with Each Interface. For each application system interface, define the responsibilities of the individuals who interact with the application system through that interface. Identify the constraints on the use of each interface that must be enforced if security is to be preserved[5]. Consider the likelihood of errors occurring in the use of the interface and identify their consequences. Consider the consequences of deliberate misuse of the interface. Without appropriate controls, deliberate misuse can even cause the execution of functions that are entirely unrelated to the intended use of that interface. Identify the management and administrative controls that will be available to ensure that the interface is used properly. Are the requirements on those using the interface and on their managers realistic? See [FDICO 77] for a general but very extensive example.

6.1.3 Separation of Duties. Examine the interfaces to ensure that security exposures will be minimized even if an interface is misused. Ideally, any action that could result in serious harm should be checked or approved from an independent interface. For example, no one individual should be able to trigger a disbursement of funds without some effective form of independent checking or approval. (See sec. 7.2.2.)

Well-planned separation of duties can greatly reduce the life-cycle costs for the background investigations that are required by OMB Circular A-71, TM1, paragraph 4.b. When there is effective independent checking on the actions of an individual, the level of personnel screening can be greatly reduced. Furthermore, independent checking may be more effective than background investigations to deter fraud.

6.1.4 Sensitive Objects and Operations. Identify the data objects to be processed—include input data, stored data, and output data[6]. Determine the sensitivity and asset value of the data objects. Identify the operations or functions that users will perform on this data. Define security requirements for each data object (or group of related data objects) with respect to the objectives of data integrity, confidentiality, and availability. Define requirements with respect to each objective to protect against both accidential and deliberate threats. If some operations such as updating a critical value are significantly more sensitive than other operations on the same object, then distinguish the different operations to be performed on each data object.

[5] While this paragraph assumes that a person is on the other side of the interface, similar comments apply when the system interacts with other automated systems. It is usually poor security practice to assume that other automated systems are trustworthy. Whenever possible, it is better practice to determine the degree to which they are untrustworthy or to take the necessary action to make them trustworthy.

[6] The data objects considered at this point are objects seen by the end users, not all the files or data sets as seen by a programmer.

6.1.5 Error Tolerance. Determine the application's error tolerance by taking into consideration the expected reliability and validity of the data and the intended objectives of the application. For example, funds disbursement or electronic funds transfer systems may have a low tolerance for data errors since such errors directly translate into dollars. Real-time control systems such as air traffic control have virtually no margin for error since human lives may be lost. On the other hand, some management information systems, particularly those used to predict future demands and resource requirements, may not be as susceptible to errors in data since a certain degree of unreliability is expected in the data, and since statistical estimating techniques are often used which account for variations in the quality of data by generating ranges or bounds for expected results. The application's tolerance for errors and the requirements for maintaining error levels to within acceptable tolerances must be defined in the security requirements.

6.1.6 Availability Requirements. If serious harm could result because the system is not available for use, then availability requirements must be defined. Availability requirements which are more rigorous than those typically supported by a well-managed computer installation are likely to be expensive. Depending on user needs, requirements can be stated in terms of a maximum length for a service interruption or in terms of its frequency of occurrence or as a combination of both factors.

6.1.7 Requirements for Basic Controls. For each application system interface and/or each data object, define detailed requirements for each of the basic controls from section 3.

6.1.8 Management Considerations. The requirements should not contain phrases such as "and so forth" or "and the like." The personnel doing the requirements specification are in a better position to fill in the details than the programmers will be.

All statements in the requirements specification should be specific enough so that tests can be designed that will tell whether the requirement is satisfied. For example, the statement "Data entry personnel should not be able to enter fraudulent payment vouchers" may be a very general requirement, but it is not specific enough to be a useful requirement on the software designers. Software requirements would have to specify the characteristics which can be used to identify a payment voucher as potentially fraudulent. The software requirements specification should be detailed enough to become the basis for the tests to be used during system integration testing.

While requirements definition logically precedes all other development activities, in practice there is often a substantial overlap, and many requirements must be modified or clarified during the remainder of the development effort. Management must establish procedures so that the requirements can be modified in an organized and carefully controlled way. The security specification should be kept current throughout the entire life cycle of the application and no changes to the application system should be permitted unless they either do not affect the security specification or have been approved and entered as an official modification to it.

Security specifications should be reviewed by all organizations that will be involved in the use or operation of the application. For any sensitive application, they must be reviewed and approved by the party responsible for security and by the organization's internal auditors.

6.2 Designing for Security

PROBLEMS ADDRESSED:

Inadequate security which is difficult to improve without a major redesign of the system.

Excessive costs for administrative controls that are needed only because the system design exposes critical code and data to unnecessary hazards.

The design stage is the time to make detailed decisions about how the controls of section 3 will be implemented. Nevertheless, designing for security is not simply a matter of appending controls onto a system design. Security concerns must permeate the entire design phase.

It is not necessary for every design decision to take the most secure approach. There are usually a variety of ways to achieve an adequate level of security, and it is only necessary that security be given an appropriate weight in each design tradeoff. Often a few basic design decisions such as those described below can dramatically reduce the vulnerabilities of the application and make security much less of a constraint in later design decisions.

A primary source of security problems is an excessively complex design which cannot be implemented correctly, cannot be maintained, and cannot be audited. While high performance goals may be an excuse for very complex designs, when security is a concern it is especially important to follow the age-old maxim "Keep it simple." It should be noted that performance and simplicity are not mutually exclusive design features.

For any sensitive application, a thorough risk analysis [WONGK 77], [LOSOT 79], including safeguard selection, should be performed at the beginning of the design phase to assure that appropriate cost-effective controls are integral to the system's design. The guidance in [FIPSP 79B] can be tailored for a risk analysis of an application system design.

Designs for each of the basic controls were discussed in section 3. The ideas described here apply to the overall design effort.

6.2.1 Unnecessary Programming. Terminals should be interfaced with the application system so as to minimize the danger that users can get unneeded programming capability. Users who can execute their own programs usually have the potential to bypass any security control. Operating systems attempt to prevent this, but their controls usually can be bypassed or avoided by clever user programming. While most users do not have the knowledge required to break security in this way, it is a poor security defense to rely on the user's lack of ingenuity.

One should not expect an absolute guarantee that programs cannot be entered from a terminal. A trap inserted almost anywhere in the code of the application or of the operating system and triggered by some designated event could cause data entered from the terminal to be executed as a program. Thus, the effectiveness of this control depends on the integrity of the program developers and of others who have had an opportunity to embed traps in the code; nevertheless, an attempt to reduce the number of people who can enter programs is important because it reduces an otherwise almost uncontrollable exposure.

6.2.2 Restricted User Interfaces. User interfaces should be tailored as specifically as possible to fulfill the user's requirements. Unneeded flexibility makes it more difficult to train users and to get them to accept the system. Such flexibility also hurts security. The greatest danger occurs when users are given unnecssary access to a general purpose programming language; however, even for a generalized query and update language, it may be very difficult to analyze all the possible ramifications on security for all the possible user actions. Once the users' actual needs are understood, interfaces should be designed to meet these needs as simply as possible with no unnecessary capabilities that complicate things both for the users and for the security analyst.

6.2.3 Human Engineering. To preserve security and integrity, user interfaces must be designed so they are easy to understand and use. This can forestall many user errors, and it decreases the chances that users will neglect or bypass controls which they view as cumbersome and annoying. This task is difficult and deserves careful thought.

System responses to the user must be clear and presented in terms of the user's view of the information—not in terms of internal details intelligible only to a programmer. Error messages should be tactful and helpful.

The user should not be required to enter excess information or answer questions repeatedly. When possible, appropriate security should be provided by default options. On the other hand, user requests that could have unusually serious effects should not be executed without having the system explain its interpretation of the request and ask the user for confirmation.

There should be a user manual for each user interface. When possible, help in using the system should be available on-line—a user should be able to type some standard request such as "help" and receive a description of the actions that are valid in the user's current context.

Human engineering may add to design costs, but if it is done effectively, it should decrease life-cycle costs.

6.2.4 Shared Computer Facilities. It is easier to protect the code and data of the application system if it does not share computer facilities with other applications. It is especially useful to exclude all program development activities from the machine that runs the application—including the development and maintenance of the application's programs themselves. For applications that run constantly—and especially for very sensitive applications—it is often cost effective to run the application on its own computer(s). In terms of simple hardware performance, small computers are now at least as cost effective as larger machines; however, the costs for system software, telecommunications, peripheral equipment, and facility management must also be considered before making a choice between dedicated small machines and shared large machines.

6.2.5 Isolation of Critical Code. The code and system data that is critical to security should be well identified so it can be more easily audited and protected. When possible, security controls should be isolated in modules that have few interactions with the rest of the application software. This makes it easier to audit these modules and protect them from unauthorized modifications during operations. Sometimes automated controls can be used to protect these modules:

* Checksums on the object code can be used to try to detect unauthorized changes.
* Hardware protection states or protection domains can protect code and data critical to security [LINDT 76C].
* If the security-critical code always resides in a fixed area of memory, then read-only memory can be used.
* Recent experimental languages, called type safe languages, may soon be available. Compilers for these languages can protect modules and their data from unexpected interactions with other modules [LINDT 76A], [WULFW 76], [LISKB 75], [LAMPB 77B], [AMBLA 76], [AMBLA 77].

Identification and partial isolation of critical code and data are reasonably easy; however, rigorous isolation is more difficult than it sounds. Without very careful planning, all system code and data will end up being relevant to security because errors or deliberate traps elsewhere can still cause security failures. This other software includes:

* The operating system and other software that supports any of the security functions.
* All parts of the application system needed to guarantee that the security controls are invoked at the appropriate times.
* Compilers and other software used to develop and maintain any security-relevant software.

When the code relevant to security is rigorously isolated from the bulk of the software, it is called a security kernel. Security kernels that protect data from unauthorized disclosure are feasible, but they require specially designed operating systems [WULFW 74B], [BRINP 70], [ARMY 78], [BERST 79], [DODKS 78], [FEIER 79], [LINDT 76C], [MCCAE 79], [POPEG 78A], [POPEG 79B], [SCHIW 75], [WOODJ 77].

6.2.6 Backup and Recovery. With appropriate contingency planning (see sec. 7.6), the services of a computer application can usually be restored within a few hours after a failure. If availability requirements are more rigorous than that, then automatic backup and recovery mechanisms may need to be included in the application software. While such mechanisms are beyond the scope of this guideline, several of the references and additional readings listed at the end of this document contain discussions of this area.

6.2.7 Use of Available Controls. The operating system and the facility management may already provide a variety of controls such as:

* User identity verification.
* Authorization for access to system files.
* Journalling of operating system activities.
* Backup and recovery procedures.

While the application system usually needs to supplement these controls, available controls should be used to the fullest extent possible. In too many cases, controls are needlessly reimplemented because controls that are already available are not understood or not utilized.

Since the security controls of an operating system are not absolutely reliable, the application system should use some data integrity checks to try to identify whether critical data has been altered; however, in general, there is no reason to believe that controls implemented in the application system will be any more reliable than those already provided by an operating system.

6.2.8 Design Review. The design review [FAGAM 76A], [FAGAM 76B] is the last reasonable opportunity to identify weak points in the security plan. Omissions or inadequacies in the security design which are not identified at this point will require costly software modifications.

The entire security plan for the application should be reviewed by a group of experts who were not part of the design effort. It is especially necessary to review all decisions which cannot be easily altered during the expected life cycle of the application.

6.3 Programming Practices for Security

PROBLEMS ADDRESSED:

Programming errors that affect security, especially flaws in the implementation of security controls.

Fraudulent additions or changes to the application code by programmers.

Programming practices [WIRTN 73], [DAHLO 72], [WEING 71], [VAND 74], [KERNB 74], [YOURE 75] that are based on the best software engineering methods are essential to security since programming errors are a common cause of serious losses. Furthermore, programming errors or deliberate traps inserted into the code can often be exploited to carry out a fraud or embezzlement. Since program errors and traps are difficult to detect, the most effective approach is to prevent their occurrence.

During the programming stage, the following practices are recommended to enhance the security of the application system.

6.3.1 Peer Review. After a programmer has completed a section of the code, review by one or more peers is requested. Typically, the reviewers will verify not only that the code does not contain any security errors but also that it:

* satisfies all design specifications,
* is efficient, and
* is easily maintainable.

Peer review is most effective when at least one other programmer is expected to review the code to the point of understanding it completely and being made equally responsible with the original programmer for any errors that remain. This is the most effective technique for preventing a programmer from making fraudulent modifications to the code. Peer review is most practical when programs are coded using programming methodologies such as those described in [FIFED 77].

6.3.2 Program Library. The program library catalogs and controls access to all versions of program modules as they are being developed. These control functions can be carried out by either manual or automated means.

The program library can provide the following types of security controls during the programming stage:

* permit only authorized persons access to program modules,
* record all accesses (especially modifications) to program modules,
* associate control data, such as record and byte counts, with program modules to facilitate detection of changes, and
* enable comparison of current versions of modules with previous versions to identify code that was changed.

More rigorous controls are needed as the program modules near completion, especially once review and testing has begun.

6.3.3 Documentation of Security-Related Code. Program documentation is needed during any software development; it is especially necessary for security if software security controls are to be reviewed for effectiveness and if they are to remain effective after software maintenance activities. Security-related modules or sections of code must be clearly identified and completely documented. By security-related code is meant:

* code that implements security controls;
* code that performs critical processing (e.g., check disbursement, real-time control); and
* code that has access to critical or sensitive data during its execution.

6.3.4 Programmer Association with Operational System. When possible, programmers should not be in a position to receive benefits from the system when it becomes operational. The temptation for programmers to make unauthorized modifications or insertions to the code will be greater if it will be easy for them to capitalize on these insertions at a later time. Programmers excluded from access to the operational system can still conspire with someone else to receive a benefit, but the conspiracy increases their risk of being detected.

6.3.5 Redundant Computation. Critical computations can be checked by redundant processing to verify the correctness of the result. The effectiveness of such checks depends, to a large extent, upon the programmers' experience and expertise and on the programming practices being used. Examples of local redundancy checks include:

* recalculation of a critical result by an alternate method;
* checking a calculated result for reasonableness and consistency with other data items; and
* examining extra attributes in a retrieved data item to assure that the data item found was the one that was searched for.

6.3.6 Program Development Tools. The choice of the programming language and of other programming tools can enhance security by increasing the reliability and correctness of the final products. Proper selection and utilization of such tools will help prevent programming errors from entering the source code. Some specific program development tools include:

* *High Level Programming Languages.* Programming languages are especially useful if they support structured control flow, extensive data definitions facilities, strong type checking, restricted scopes for program variables, and well-defined module calling conventions. Compilers for such languages can do extensive checking to identify program errors [WULFW 76], [LISKB 75], [LAMPB 77B], [AMBLA 76], [AMBLA 77].

* *Preprocessors.* Many of the advantages of high-level programming language can be accomplished through a preprocessor. A preprocessor accepts programs written in an improved dialect of a language and translates them into the language. Preprocessors can be used to:

  - eliminate some of the more restrictive conditions in an existing language (e.g., allow structured flow in FORTRAN programs), and
  - provide automated quality control by checking that program modules meet the coding standards of this project.

* *Other Tools.* Program development tools such as those that reformat source code, produce cross-reference listings, and aid debugging are useful to help programmers manage the complexities that they must deal with.

6.4 Test and Evaluation of Security Software

PROBLEMS ADDRESSED:

Errors and omissions that affect security, especially flaws in the implementation of security controls.

Test and evaluation [LARSR 75] attempts to demonstrate that a system is:

* reliable,
* meets its specifications, and
* meets the requirements of the user.

Careful and thorough testing and evaluation can improve system security by uncovering errors, omissions and other flaws in the application system's design and code. However, even careful and thorough testing and evaluation cannot assure that all flaws in the system will be found. Security requirements are usually stated in terms of events that must not occur under any circumstances. Testing is less effective in evaluating these negative requirements than it is in verifying the positive performance of the system under typical circumstances. Formal proofs of correctness are able to evaluate whether very general and even negative requirements are fulfilled; however, formal proofs are too difficult for widespread use at this time.

If security has not been considered throughout the development life cycle and if attempts to avoid flaws have not been made from the beginning, then the elimination of flaws identified by test and evaluation cannot be expected to improve security significantly.

6.4.1 Test Plan. A test plan for security should describe what is to be tested and what testing methods or tools will be needed. It should include tests that identify the system's response to abnormal, unusual, improbable, and illegal circumstances that may exist during both data input and processing.

The importance of testing for exceptional conditions is illustrated by the case of an accounts payable system where the software security control was that for every payment made, a record of the payment was sent to the appropriate regional office of the organization. The regional office was responsible for checking the payments manually. In this case, an appropriate administrative control was thought to exist and, thus, software responsibility was minimal. However, it was learned that a request for payment with a nonvalid (null) regional office identifier was still processed and a check issued.

The documentation of requirements, if it has been kept up to date, provides the basis for formulating the test plan.

6.4.2 Static Evaluation. These techniques, which involve examination and analysis of the systems documentation and code, represent the most effective way to detect deliberate traps or other unauthorized modifications. However, due to the complexity of most systems and the limitations of automated techniques and tools [HANTS 76], it is not currently practical to analyze systems completely using static evaluation methods. In addition, static evaluation does not examine the system in a "live" or operational mode so that errors in the execution environment are not detected. Specific techniques and tools include:

* *Code Review.* Portions of the source code are evaluated to determine if they implement the design specifications and are free from errors. In most cases, it will be impossible to review all of the system's code. Generally, samples of the code will be reviewed—especially critical modules or critical portions of the code. The code review can be done by an internal test and evaluation team that is involved in the system development or by an independent (third party) team—either internal or external to the organization. Code review differs from peer review, discussed in section 6.3.1, in that the code review is performed by individuals who were not involved with the design and programming of the application.

* *Penetration Studies.* A few individuals can be challenged to find unknown weaknesses in the security controls. If a penetration study [LINDR 75], [LINDT 76B], [HOFFL 77], [HSIAD 79] is to produce useful results, the context within which the penetrators are to work must be clearly defined. Penetration studies to identify programming errors can be expensive and are useful only if someone believes that there are no remaining errors that can affect security. The failure of a penetration study does not demonstrate that the application is secure nor does success mean that the application is totally insecure in any practical sense.

* *Source Code Analyzers.* These software tools (LYONG 74] aid the evaluation process by providing details about specific characteristics of the source programs. Examples include:

  – cross reference listings are an aid during code review and may be useful to identify "suspicious" variables and source statement references.

  – the variables which can influence control flow decisions can all be identified or variables which could be read before any value has been assigned to them can be identified.

6.4.3 Dynamic Testing. These techniques involve executing the application system, or portions of the system, with test data and comparing the actual results with expected or known results.

Dynamic testing may not detect unauthorized modifications or insertions to the system's code since they can usually be concealed by a clever programmer. However, program analyzers (see below) can provide some support in this area.

During dynamic testing, production data files should not be used as test data. In addition, all other important data files should be taken off-line or otherwise protected during the testing period to eliminate the risk of their being destroyed or their contents inadvertently revealed.

Dynamic testing is only practical for selected test cases. Fundamental questions such as, "Which test cases should be chosen?" and "How many test cases are enough?" must be answered. The answers to these questions depend upon the system being tested and upon the experience of the testing team.

The following tools can be used to aid the dynamic testing process:

* *Program Analyzers.* A program analyzer is a computer program that collects data about another program's operation while that program is executing. For example, an analyzer can determine whether test inputs have caused all of a program's statements to be executed, or it can maintain a record of the number of times each statement of a program was executed. Program analyzers can be particularly useful for evaluating how thoroughly the test data has exercised the program being tested. In addition, they can be used to identify extraneous code that might be an unauthorized insertion.

* *Flaw Hypothesis Method.* Security flaws can be hypothesized [LINDR 75], [LINDT 76B], [HSIAD 79] based on analogous flaws found in other systems and then tested for existence on this system. This is an effective approach for selecting test cases that are likely to find security flaws.

7. PRESERVING SECURITY DURING OPERATIONS

Following development and successful acceptance testing, the application will enter the operations phase (i.e., its production environment). Even though the primary controls that were imbedded in the application will begin to function and provide the expected protection, security measures should not be relaxed during this phase. There are a number of procedures and practices which can be used both to supplement and to complement software security controls [IBMCO 76]. These include protection and control of data during input preparation, off-line storage and dissemination of output; employee hiring, job assignment, termination and security training; responding to security variances; software modification; hardware maintenance and advance planning to ensure continuity of critical ADP services at all times.

7.1 Control of Data

Protection, control and integrity of data is a continuing necessity throughout the stages data passes through in an automated application. Primary controls such as data validation and authorization that protect on-line data were discussed in section 3. This section deals with the controls needed during the various stages of input preparation, data media storage and output dissemination.

7.1.1 Input Verification. Input verification is the process of assuring accurate transcription of source data to machine-readable form. In most cases, the transcription process is accomplished by manually keying the source data at an input device such as a keypunch machine or a terminal with a CRT display [JANCE 74], [MARTJ 73], [RCBCG 78C].

PROBLEMS ADDRESSED:

If not detected, errors made during source data transcription can be propagated through the data base by additional processing of that data. The resulting loss of data integrity can seriously affect agency operations or management decisions that use this data.

Processing erroneous data can cause unanticipated control flow paths to be taken in the application software leading to program halts or bringing the application system down.

Specific Techniques:

* Visual Verification. The keyed data is visually compared to source data. For visual comparison, the keyed data may be in the form of a punch card with printed characters recorded on it, a human-readable listing, or a display in a CRT. Since visual verification is time consuming and subject to human error, it is only practical when source data is entered on-line for immediate use and where input validation controls exist.

* Key Verification. The source data is keyed a second time and automatically compared with the transcribed source data previously keyed. Detected errors are then corrected, rekeyed and reverified. While key punch verifiers are the most common devices used for key verification, key-to-tape and key-to-disk devices can also be used for this purpose and have the additional advantage of recording the source data on a high-speed storage medium.

* Check Digit. A source data item contains an extra check digit which is derived from all the other characters of the item by a special algorithm. When the source data item is keyed, hardware in the key device calculates the check digit and compares the result with the check digit that was just keyed as part of the source data item. If they are the same, it is assumed that the keying operation was correct. In general, the check digit is not calculated when keying data but rather is part of the predetermined alphanumeric code for the item being keyed. Thus, check digits apply best to data items whose values range over a known set of values which are not likely to change frequently (e.g., part number, employee number, department number) [JANCE 74]. Check digits are particularly useful when entering data on-line or when verifying data read by optical character readers. They can also be used to validate data input (see sec. 3.1.2). Although this technique is effective in most cases, it is not infallible since it is possible to make compensating keying errors in a source data item which do not alter the check digit.

* Control Totals. To use this technique, the source data is divided into small groups or batches of the same type of work. Individual totals are monitored for important items (or all items if desired) in the group and these totals form an input source record which is keyed along with the group. After the keying process, the keyed items are retotaled and the results compared with the source record containing these totals. A discrepancy

indicates a mistake in the keying operation or a mistake in calculating the original total. This technique will require processing capability to reevaluate and compare the control totals. In addition, there is a trade-off between the size of the group and the ease of locating the errors since the technique detects an error in a group but does not indicate in which record of the group the error occurred. In an extension of this technique, known as cross-totals, a group consists of a number of records, each containing the same type of items. If one thinks of the group as a matrix, specific (or all) items in the group can be totaled by row and by column and the row and column totals keyed with the rest of the group. After the keying process, new row and column totals are calculated, and compared with the originals. Unless compensating errors have been made, keying mistakes will cause some of the new row and column totals to differ from the original row and column totals. For each pair of new row and column totals where either one or both of the row or column totals differs from the original, the data item at the intersection of the corresponding row and column will be in error.

* Machine Readable Source. Devices which read source data directly, such as optical character readers [FIPSP 76C], [RCBCG 78C], eliminate the time and expense of key transcription and verification. Used in conjunction with check digits, this technique provides a powerful means of verifying the accuracy of source data entry.

Management Considerations:

Input verification is a costly and time-consuming process. It is only useful for detecting keying errors and even when the keyed data is also verified, there is still a possibility of error.

7.1.2 Data Storage Management. Data storage management [IBMCO 76], [CARRJ 77], [BECKR 77] deals with the maintenance of and access to data storage media.

PROBLEMS ADDRESSED:

Data residing on storage media are subject to accidental and intentional modification, destruction, and unauthorized disclosure. An agency's mission could be disrupted if its critical master files or backup and recovery data were altered or destroyed.

Specific Techniques:

* Controlled access to storage area. Data storage media, when not in use, should be located and maintained in an area to which physical access can be controlled. The only person who should have access is the one having custody.

* User authorization. Media should be released only to individuals who have specific authorization for their use. Authorization for access to certain sensitive data should not entail access to all sensitive data.

* Accounting procedures. The movement of data storage media outside the off-line storage area should be accounted for at all times in a log which includes, at a minimum:

- identifying information,
- destination,
- authorized signature, and
- estimated time of return.

* Backup of sensitive data. The surest form of protection against destruction and alteration of data is the maintenance of additional copies at another location. Deciding which data to backup in this manner is a matter of management policy, but once that decision has been made, the creation and removal of backup data offsite should be handled on a routine schedule.

* Data encryption. (See sec. 3.6.2.)

7.1.3 Output Dissemination Control. Controlling the dissemination of the output of programs processing sensitive data helps to assure continued protection for the data after it leaves the computer.

PROBLEMS ADDRESSED:

If output is not carefully handled, but merely left to users to remove from the printer and separate, it is very easy for them accidentally to pick up more than they should. Even if printout is carefully separated, because of the similarity of most computer output, it could be confused. Unless there is some sort of accounting known to users, there could be a tendency to dispose of unwanted output or even to use like-appearing output for the wrong purpose.

It is much easier to steal information in hard copy form than out of a computer or from storage media. Therefore data is particularly vulnerable at the output dissemination stage. The following techniques provide some protection:

* Logging receipts. Limiting the number of copies, numbering each copy, and requiring an authorized signature for each copy is the only absolute way of assuring that copies of output go to those for whom they are intended.

* Distribution by mail. Distributing output through a mail delivery system is another, though less certain, way of assuring that copies go to those for whom they are intended.

* Labeling. Automatically marking each page of sensitive output with the appropriate sensitivity classification will provide a warning to someone who is not authorized to receive a copy.

7.2 Employment Practices

There are a number of practices, most as valid in manual processing environments as in automated ones, which can serve to lessen the vulnerability of a system to harmful acts perpetrated either accidentally or maliciously by employees [IBMCO 76], [CARRJ 77].

PROBLEMS ADDRESSED:

Employees who deal directly with data are in a position to reveal it to others as a curiosity or for personal gain or for revenge. They are in a position to modify data to the data subject's benefit or detriment. They may also accidentally make mistakes which can affect the subject of the data or the organization adversely. In addition, employees who deal with financial or other valuable data files are in a position to introduce information which will make either themselves or others beneficiaries of the system.

Specific areas of employment practices that affect the security of an application include employee hiring, assignment of responsibilities, and termination.

7.2.1 Hiring Procedures

* Investigation of Criminal Record. The existence of a criminal record may, or may not, indicate that a person is dishonest. The risk an organization is willing to take in this area will depend on the sensitivity of its operation and the duties to be assigned to the employee.

* Checking References. While poor references can provide ample justification for not hiring an individual, good references should not be the only criteria for employment. Characteristics, such as integrity, are not necessarily invariant over time. People who have been honest all of their lives can become dishonest if the conditions are right. With this understanding, employment, character and credit reference should be checked as thoroughly as possible. Employment references can be used to determine an individual's competence to perform, or to be trained to perform, the tasks required on the job. Character references can be used to determine such things as trustworthiness, reliability, and ability to get along with others. Credit references can be used to determine a person's financial habits which can, in turn, be an indication of maturity and willingness to accept responsibility for one's own actions. The temptation of working with a system which could produce financial benefits might be more than an individual in financial straits could resist. Such a person would be easy prey for external subversion. Credit references can also be an indication that people in financial difficulty will need to take a great deal of time away from the job, or will spend a lot of time on the job taking care of their own affairs.

* Security Clearances. The Federal Personnel Manual System FPM Letter 732-2 sets forth the requirements for security clearances for personnel associated with computer systems in the Federal Government. Complying with the requirements of the FPM Letter will result in observing the above hiring procedures.

7.2.2 Assignment Procedures

* Assignment in writing. All tasks assigned to an individual should be enumerated and specified in writing, and the employee should be asked to acknowledge understanding of the assignment in writing.

* Separation of duties. The precedent for separation of duties comes from the pre-automation era, but that is not an excuse for abandoning this well-established concept now that most operations are automated. It is still a very effective risk reduction technique. An example of such separation, originating in manual operations but continuing into automatic operations, is the segregation of the responsibility for preparing payment vouchers from the responsibility for approving payments. The responsibility for each of the functions listed below should, to the extent possible, be assigned to separate individuals. If such separation is not possible, an attempt should be made to separate closely related functions such as applications programming, program design review, program test and evaluation, and program maintenance. It is also important to define each function clearly so that there will be no overlap in responsibility from one function to another.

- Data collection
- Data preparation
- Data entry
- Data base administration
- Authorization to use data for specific purposes
- Custody of automated data
- ADP operations
- Systems programming
- Applications programming
- Program design review
- Program test and evaluation
- Program maintenance
- Security of ADP facility and equipment, including remote terminals
- Data communications
- Output dissemination
- Privacy requirements not covered above
- Internal audit
- Security coordination and planning

* Acting in concert. A requirement for having two or more individuals performing distinct parts of the same task, usually within a specified time limit, is a further refinement of the concept of separation of duties. It requires certain individuals, in separate locations or possessing separate keys or having separate means of access to the same resource, each following certain assigned procedures independently of the other individuals concerned, so that the culmination of their activity will be the desired event. For tasks that are susceptible to fraudulent or other unauthorized activity, the probability of such activity successfully occurring is reduced when collusion between employees is required.

* Enforced vacations. It is well known that vacations contribute to the health of employees by relieving the tensions and anxieties that typically develop from long periods of work unbroken by extended time off. Tense, anxious employees are more likely to have accidents or make errors and omissions in performing their duties. In addition, if all employees are forced to take vacations, there will be less possibility that one of them can set up a fraudulent scheme which depends on the employee's presence either for continuity or for secrecy. Even if the employee's presence is not necessary to the scheme, enforced vacations can be a deterrent to embezzlement because the perpetrator would always fear discovery during the employee's absence.

* Job rotation. Moving employees from one job to another at random intervals will have the same deterrent effect on fraud that a policy of enforced vacations will. There will be the additional benefit of having employees who can take each other's places in cases of illness or termination.

* Account limitation. Placing upper limits on the amount of funds which can be handled by an employee or billed to a project account and on the computer time that may be used by a project account will deter attempts by employees at embezzlement as well as their attempts to use computer time for their own use or to sell it as a commodity.

* Limitation of physical access. Physical access to a computer facility should be carefully controlled by barriers so that an individual who is there for only casual reasons such as submitting a job or picking up output cannot move freely around the area. In particular, programmers, who usually feel they have a right to be there, should be denied access to the computer room.

* Limitation of data authorization. Access to data should be granted with deliberation of each individual's need for such access, rather than according to rank or position or precedent. Group authorizations should be avoided except when there is adequate justification.

* Observation by supervisor. Neither the fact that an employee has a security clearance nor the fact that the employee is a long-time trusted employee should obviate the need for continuing observation by a supervisor who knows the employee well and who is able to observe signs of stress or aberrant behavior.

7.2.3 Termination Procedures. These practices should be observed upon the departure of any employee regardless of the conditions or mood of the termination. Under certain conditions, when the employee is likely to develop feelings of resentment, it is probably better to grant severance pay and terminate the employee's contact with the organization immediately. In other cases, when the parting of ways is on friendly terms, the authorization necessary for the employee to wind up the job can be continued in effect.

* Revoking all authorizations. All authorizations which have been granted to a departing employee (e.g., authorization to sign for sensitive data, authorization to access certain data files) should be revoked, a letter to that effect should be written and the employee's name should be removed from all lists of such authorizations. If the departing employee's authorizations included the granting of authorizations to others, these should also be reviewed. In most cases, there will be no need to revoke such authorizations, but some agency's regulations may require that such authorizations be granted under the signature of the responsible individual. However, it is possible that an authorization review may uncover unnecessary or erroneous authorizations due to changes in job functions of employees without corresponding corrections to the authorization data.

* Retrieving all keys, badges, etc. All keys, badges and other devices which are used to gain access to premises or information or equipment should be retrieved from a departing employee.

* Changing locks. The combinations of all locks known to a departing employee should be changed immediately. Periodic changes, which should also be performed, are not sufficient.

* Deleting passwords. Not only should departing employees' passwords be revoked, they should also be removed from the list of effective passwords so that the deleted password cannot be reused.

* Reviewing employee's obligations to the organization. If there are special conditions to the employment, such as denial of right to use certain information, these should be reviewed with the departing employee and a letter to that effect should be acknowledged by the employee in writing.

7.3 Security Training

Security training refers to the training that is necessary to indoctrinate personnel in the performance of assigned security tasks and to security awareness training [FIPSP 74], which addresses the possible threats which can occur, what to do about them, and how to prevent them. While such training might seem to deal with the obvious and self-evident, every organization which has instituted such training reports very favorable results.

PROBLEMS ADDRESSED:

When employees are not informed of management's interest in security, they are not likely to take steps to prevent the occurrence of violations.

Employees cannot be expected to assume responsibility for assuring security unless they are trained in the steps to take and encouraged to do so.

7.3.1 Task Training. Task training in specific security responsibilities should include all the skills and detailed outlines of the steps to take to adequately perform security related duties, such as operating fire extinguishers or notifying management or restarting the computer after a power failure.

7.3.2 Security Awareness Training. This should be a part of the indoctrination for all new employees whether or not they also are assigned specific security responsibilities. It should also be a continuing effort, addressing new subjects as the state of the art progresses and including periodic reviews of the need for security. Initial topics should include management's emphasis on security, universal responsibility of all employees, data sensitivity (privacy, confidentiality, etc.), and the effect of delays in operations on the agency's mission. Later sessions could include such things as physical security, variance reporting and response, and contingency planning but should never become so technical that the audience will be lost.

7.4 Responses to Security Variances

The possibilities for security variances to occur in the operations area are multitudinous and, therefore, very difficult to describe. They can range from an unauthorized individual attempting to get into the computer room, to a fire extinguisher that does not work, to a door that has been propped open for convenience, to faulty water sprinkler heads. Anything in conflict with existing rules or anything that cannot be explained or anything that is otherwise unusual should not be ignored but should be reported immediately to a supervisor or other designated individual to minimize risks to the organization and to the safety and well-being of the employees.

There should be a means of reporting such situations in confidence and without the stigma of "whistle-blowing." The procedure should be simple and should not encumber the reporting individual or tie up the employee's time in the follow-up. The need for such reporting should be stressed in awareness training and examples presented of what could happen if certain occurrences passed without notice, such as failure to check the credentials of an employee attempting to enter a restricted area merely because the checking individual is familiar with the employee and knows that the employee has had the appropriate authorization in the past. If the procedures for variance reporting are convenient and employees are continually encouraged to make use of them, the follow-up of incidents can be revealing to management and lead to the establishment of workable procedures for correcting potentially serious security situations. The emphasis in reporting variances should be on the circumstances rather than on the people who take advantage of them. In spite of management's encouragement to employees to report security variances, management should not expect employees to cooperate completely, particularly when such cooperation may require employees to report on the activities of their friends and co-workers.

7.5 Software Modification and Hardware Maintenance

Software modification and hardware maintenance [CARRJ 77] are necessary to the continuing usefulness of any system, but it should be recognized that performance of these activities increases the risk that errors, accidents, and intentional acts can occur.

7.5.1 Modification of Software

PROBLEMS ADDRESSED:

While changes or additions to code are being made, errors could be admitted to a program which could destroy data, or modify it to the extent that it would not be usable, or code could be inserted to perpetrate fraudulent activity.

The same programming practices for security described in section 6.3 above apply when modifying software in an applications system. In addition, observing the following practices will provide additional protection for the system while making the changes.

* Code reconciliation. New code to be inserted should be compared to code being replaced and to specifications for the change.

* Availability of old code. The code which is being replaced should be kept readily available in the program library for a reasonable period of time (at least 4–6 weeks) in case trouble develops in using the new code.

* Documentation. All changes should be thoroughly documented, including the intended result, the reason for ordering the change, any special effect on data or output and, especially, any change produced in configuration.

* Use of test data. All changes should be subjected to testing for predicted and desired results but production data should not be used for this. Special test data which simulates production data should be maintained and updated for testing purposes. A copy of production data can be used as test data if it does not contain sensitive information.

* Use of original tests. The modified software should be subjected to the same testing as the original software. The tests described in section 6.4 should be repeated on the modified software and the results compared.

7.5.2 Maintenance of Hardware

PROBLEMS ADDRESSED:

During the time hardware is being maintained, any data which is on-line could be inadvertently destroyed or sensitive data could be revealed to maintenance personnel.

Specific Techniques:

* Removing live data. All production data should be taken off-line for the duration of maintenance.

* Clearing memory. The main memory should be zeroed to remove all traces of data and applications software.

* Regular Preventive Maintenance. Preventive maintenance should be performed on a regularly scheduled basis rather than on an "as needed" basis in order to prevent the kind of errors which stem from poor maintenance.

7.6 Contingency Planning

Contingency planning [CANNR 72], [FIPSP 74], [IBMCO 76], [CARRJ 77], [BECKR 77], [MARTJ 73], [SCOML 75] consists of the advance plans and arrangements which are necessary to ensure continuity of the critical functions of an application system. The plans should cover all events of total or partial cessation of operations or destruction of the data base or physical facility. Such planning should include procedures and availability of equipment for both automated and manual procedures.

OMB Circular A-71, Transmittal Memorandum Number 1, July 27, 1978, requires all Federal agencies to develop and maintain "appropriate contingency plans." Viewed from one perspective, contingency planning is a valuable computer security safeguard which can be selected as a result of a risk analysis to satisfy some of the agency's protective needs. In fact, a risk analysis should be the basis upon which an agency determines the levels and extent of contingency planning "appropriate" to its situation. Consequently, the following discussion of the kinds of planning relevant to application system emergencies assumes that such planning will be initiated following the performance of a risk analysis of the specific application system.

PROBLEMS ADDRESSED:

An agency can be prevented from carrying out its mission in a timely manner due to the inability to process data or to recover information vital to management decisions.

The two situations to which application systems are generally vulnerable are loss of processing availability and damage to data. It follows that the kinds of contingency planning required are:

1. what to do if the network or computing system serving the application is down, and
2. what to do if the data base becomes so fraught with errors as to preclude further use without compounding the damage.

In both instances, the criticality of the mission supported and the urgency of the requirement for system output must be considered. The less delay a system can countenance, the more extensive will the planning need to be. All plans must be carefully and extensively documented, kept current, and disseminated to all personnel who will be affected by them. Personnel must be trained in their assigned tasks and plans must be tested for feasibility.

The following kinds of planning are relevant for application system emergencies:

7.6.1 Identification of Critical Functions. The functions in an applications system which are critical to an agency's mission should be identified by performing a risk analysis, which will indicate not only which functions are immediately critical but also the time periods after which other functions may also become critical.

7.6.2 Alternate Site Operations. Another organization should be found which uses the same or compatible equipment and, which, during an emergency, would be able to allot time for critical functions. Such an arrangement will usually be reciprocal. It should be effected on a formal basis between the organizations involved, not left to a friendly arrangement between individuals. Planning should extend to ensuring the readiness at the alternate site of up to date, tested software and sufficient preprinted forms and any other special supplies (envelopes, stamps, binders, display devices, storage facilities, etc.) to last through one reorder cycle. In addition, a method must be devised for communicating with the alternate site (telephone lines, messenger, etc.). Adequate transportation facilities may have to be provided and specific personnel must be designated and trained.

7.6.3 Manual Replacement of Limited Processing. It may be possible that processing of critical functions can revert to the procedures used prior to automation of the application. Certain types of data are amenable to manual processing for short periods of time; for others, only a severely streamlined version of the function can be accomplished. When continuity is important, any special equipment required for manual operation must be close at hand. Necessary particulars include working space, availability of timely data, page copies of source data, conversion of data to manual use, special equipment, preprinted forms, communications, transportation, selection and training of personnel, and a process for recording all manual transactions.

7.6.4 Backup of Data. The easiest way to ensure against loss of data is to keep additional copies in readily accessible and/or secure places. Decisions concerning how many copies to keep, where to store them, and how long to keep them, will depend on the criticality of the application, the vulnerability of the data, and the size of the data bases.

7.6.5 Recovery of Data. Sometimes data which has been either accidentally or intentionally modified will be so altered that further transactions will only compound the damage. In order to meet such an eventuality, a list of key personnel for each data base should be prepared so that they can be summoned without delay when needed. The individuals concerned would probably be senior systems programmers and senior applications programmers. In addition, the list should include the security officer or someone else who could authorize the substitution of extraordinary procedures if necessary, such as access to the computer room if not ordinarily granted, access to computer resources not ordinarily authorized and use of unapproved programs on live data. All personnel who might be involved in such a situation, including the computer operator, should be instructed in emergency measures in advance. Lists of such personnel should be kept up to date and should contain names of alternates as well. A priority, based on experience, should be established for checking the source of the errors and a small data base should be maintained for testing recovery procedures.

7.6.6 Restoration of the Facility. If a computer facility is destroyed or damaged, there must be not only plans for continued operation, but also plans for the restoration or relocation of the facility. Recovery from destruction can include locating space or real estate; building or remodeling existing buildings; procuring hardware, supplies, office furnishings, and ancillary equipment; and transferring operations from the emergency site to the permanent one. Such plans will normally be the direct responsibility of facility management but applications systems management may well be called on for support especially in the areas of workload planning, space requirements, equipment requirements and moving operations. Advance preparation and periodic updating of the information necessary to these activities will ensure its completeness and timeliness.

Management Considerations:

* Even if contingency plans are documented, kept up to date, disseminated, and tested, and personnel are well trained, there may still be unanticipated or unfamiliar situations that require an emergency response in a time frame that is difficult to respond to adequately. Recognizing this, management should require that personnel who are responsible for security in general and contingency planning in particular maintain an awareness of accidents, disasters, emergencies, etc. that have afflicted automated systems in other Federal agencies or private organizations in order to minimize the possibility of similar events occurring at their facilities. The sprinkler system accident which occurred at the Bureau of the Census in 1979 is a case in point. Line personnel responsible for reacting to emergency situations should be briefed on those experiences that are relevant and discussions should be held to ascertain how similar incidents could be handled, should they occur. At a minimum, personnel should be constantly aware of the location and purpose of all emergency control switches.

* Management should also recognize that in some situations personnel will not have time to stop and read the procedures. Even well-trained personnel who have been active in periodic testing of contingency planning procedures cannot be expected to memorize all of them. Although it is obvious that more frequent testing will improve employees' familiarity with the plan and their readiness to handle emergency situations, it is also evident that additional costs will be incurred. One of the major cost benefit considerations in contingency planning is balancing adequacy of testing against the cost.

REFERENCES AND ADDITIONAL READINGS

[AACO 78] Arthur Anderson & Company. A Guide for Studying and Evaluating Internal Accounting Controls. 1978 January. Subject File AA 2880, Item 1.

[ABBOR 76A] Abbott, R. P., et al. Security Analysis and Enhancements of Computer Operating Systems. Washington, DC: National Bureau of Standards; 1976 April. NBSIR 76-1041.

[ABBOR 76B] Abbott, R. P.; Bloone, L. W.; Morvay, I. M.; Tokubo, S. A Bibliography on Computer Operating System Security. Livermore, CA: Lawrence Livermore Laboratory, University of California; 1976 April 15; Technical Report.

[ACLUN 75] American Civil Liberties Union. The Privacy Act of 1974: What Are Your Rights. The Privacy Report. 2(5): 1-9; 1975 March.

[ACLUN 77A] American Civil Liberties Union. Privacy and Computer Security. The Privacy Report. 4(11); 2-4; 1977 June.

[ACLUN 77B] American Civil Liberties Union. Who Are You? Identifiers and Identity Documents. The Privacy Report. 4 (10): 1-9; 1977 May.

[AFIPS 74] AFIPS System Review Manual on Security. Montvale, NJ: American Federation of Information Processing Societies, Inc.; 1974.

[AICPA 77] American Institute of Certified Public Accountants, Computer Services Executive Committee. The Auditor's Study and Evaluation of Internal Controls in EDP Systems. New York, NY: American Institute of Certified Public Accountants; 1977.

[ALLEB 76] Allen, B. R. Embezzlement and Automation. Proceedings IEEE CompCon International Conference; 1976 February; San Francisco, CA. 187-188.

[AMBLA 76] Ambler, A. Report on the Language GYPSY. Austin, TX. University of Texas; 1976 August. ICSCA-CMP1.

[AMBLA 77] Ambler, A. L.; Hoch, C. G. A Study of Protection in Programming Languages. Proceedings of ACM Conference on Language Design for Reliable Software; 1977; 25-40.

[ANDEJ 72] Anderson, J. P. Information Security in a Multi-User Computer Environment. Advances in Computers. Vol. 12; 1972.

[ANDER 72] Anderson, R.; Fagerlund, E. Privacy and the Computer: An Annotated Bibliography. ACM Computer Review. 13: 551-559; 1972 November.

[APINC 78] Auerbach Publishers, Inc. EDP Auditing. Pennsauken, NJ: Auerbach Publishers, Inc.; 1978.

[ARMY 77] U.S. Department of Army. Automated Systems Security. 1977 October. U.S. Army Regulation 380-380.

[ARMY 78] U.S. Army First Automation Security Workshop. December 11-12, 1978; Leesburg, VA.
[Note: While the workshop objective was to discuss state of the art and issues confronting future Army automation security directions, much of the material in the workshop proceedings is applicable to all Federal agencies. The workshop was sponsored by the Assistant Chief of Staff for Intelligence, the Assistant Chief of Staff for Automation and Communications, the U.S. Army Integrated Software Research and Development Working Group and the U.S. Army Systems Command. The workshop coordinator was Merton J. Batchelder, U.S. Army Systems Command.]

[BECKR 77] Becker, Robert S. The Data Processing Security Game: Safeguarding Against the Real Dangers of Computer Abuse. Elmsford, NY: Pergamon Press Inc.; 1977. ISBN: 0-08-021790-7.

[BELLD 73] Bell, D. E.; La Padula, J. Secure Computer Systems: A Mathematical Model. Bedford, MA: MITRE Corporation; 1973 November; MTR-2547, Volume II. Available from: NTIS, Springfield, VA; AD 771 543.

[BELLD 74] Bell, D. E. Secure Computer Systems: A Refinement of the Mathematical Model. Bedford, MA: MITRE Corporation; 1974 April; MTR-2547, Volume III. Available from: NTIS, Springfield, VA; AD 780 528.

[BEQUA 78A] Bequai, August. Computer Crime. Lexington, MA: Lexington Books; 1978. 224. ISBN: 0-669-01728-0, $15.95.

[BEQUA 78B] Bequai, August. White-Color Crime: A 20th-Century Crisis. Lexington, MA: Lexington Books; 1978. 208. ISBN: 0-669-01900-3, $14.95.

[BERGJ 72] Bergart, J. G.; Denicoff, M.; Hsiao, D. K. An Annotated and Cross-Referenced Bibliography on Computer Security and Access Control in Computer Systems. Ohio: Ohio State University; 1972 November; Technical Report OSU-CISRC-72-12. Available from: NTIS, Springfield, VA; AD 755 225.

[BERGJ 75] Berg, J., ed. Exploring Privacy and Data Security Costs—A Summary of a Workshop. Washington, DC: GPO; 1975 August; NBSTN 876. Available from: NTIS, Springfield, VA; COM 75-11113.

[BERST 79] Berson, T. A.; Barksdale, G. L. (Jr.) KSOS—Development Methodology for a Secure Operating System. AFIPS Conference Proceedings, Vol. 48; 1979 June 4-7; New York, NY. Montvale, NJ: AFIPS Press; 1979: 365-370.

[BIGGC 80] Biggs, Charles L.; Birks, Evan G.; Atkins, William. Managing the Systems Development Process. Englewood Cliffs, NJ: Prentice-Hall, Inc.; 1980. 408. ISBN: 0-13-550830-4.

[BJORL 75] Bjork, L. A., Jr. Generalized Audit Trail Requirements and Concepts for Data Base Applications. IBM Systems Journal. 14(3); 1975.

[BOEHB 73A] Boehm, B. W. Software and Its Impact: A Quantitative Approach. Datamation. 1973 April.

[BOEHB 73B] Boehm, B. W.; Brown, J. R.; Kaspar, H.; Lipow, M.; MacLeod, G. S.; Merritt, N. J. Characteristics of Software Quality. 1973 December 28. Doc. No. 25201-6001-RU-00. NBS Contract No. 3-36012.

[BRAND 73A] Branstad, D. Privacy and Protection in Operating Systems. Computer. 6: 43-46; 1973 January.

[BRAND 73B] Branstad, D. Security Aspects of Computer Networks. AIAA Computer Network Systems Conference; 1973 April. Paper 73-427.

[BRAND 77] Branstad, D.; Gait, J.; Katzke, S. Report of the Workshop on Cryptography in Support of Computer Security. Washington, DC: NBS; 1977 September; NBSIR 77-1291. Available from: NTIS, Springfield, VA; PB 27144, $5.25.

[BRAND 78] Branstad, D., ed. Computer Security and the Data Encryption Standard. Washington, DC: GPO; 1978 February; NBSSP 500-27. Available from: GPO, Washington, DC; SN 003-003-01891-1.

[BRINP 70] Brinch-Hansen, P. The Nucleus of a Multiprogramming System. Communications of the ACM. 13(4): 238-241; 1970 April.

[BROWP 73] Browne, P. Taxonomy of Security and Integrity. In [HOFFL 73].

[BROWP 74] Browne, P. S.; Cosenting, J. A. I/O—A Logistics Challenge. Proceedings CompCon 74 Eighth IEEE Computer Society International Conference; 1974 February; 61-64.

[BROWP 76] Browne, P. S. Computer Security—A Survey. AFIPS Conference Proceedings; 1976. NCC; 1976; 45: 53-63.

[BURKE 76] Burke, E. L. Discovering Illicit Computer Usage. Proceedings IEEE CompCon International Conference. 1976 February 24-26; San Francisco, CA. 178.

[BURNK 76] Burns, K. J. Keys to DBMS Security. Computer Decisions. 8(1): 56-62; 1976 January.

[BUSHA 75] Bushkin, Arthur A. A Framework for Computer Security. McLean, VA: System Development Corporation; 1975 June. Revised edition, TM-WD-5733/000/01.

[BUSHA 76] Bushkin, A. A.; Schaen, S. I. The Privacy Act of 1974: A Reference Manual for Compliance. Santa Monica, CA: System Development Corporation; 1976.

[CAMPR 79] Campbell, Robert P.; Sands, Gerald A. A Modular Approach to Computer Security Risk Management. AFIPS Conference Proceedings, Vol. 48; 1979 June 4-7; New York, NY. Montvale, NJ: AFIPS Press; 1979: 293-304.

[CANNR 72] Canning, R. Computer Security: Backup and Recovery Methods. EDP Analyzer. 10(1); 1972 January.

[CARLJ 78] Carlstedt, J. Protection Errors in Operating Systems: A Selected Annotated Bibliography and Index to Terminology. Los Angeles, CA: Information Sciences Institute, University of Southern California; 1978 March; ISI/SR-78-10. Available from: NTIS, Springfield, VA; AD 053 016.

[CARRJ 77] Carroll, John M. Computer Security. Los Angeles, CA: Security World Publishing Company, Inc.; 1977. ISBN: 0-913708-28-3.

[CHAMD 75] Chamberlain, D. D.; Gray, J. N.; Traiger, I. L. Views, Authorization and Locking in a Relational Database System. AFIPS Conference Proceedings; 1975. NCC; 1975; 44: 425-430.

[COGOS 76] Committee on Government Operations, U.S. Senate. Problems Associated with Computer Technology in Federal Programs and Private Industry. Washington, DC: Government Printing Office; 1976 June. 448. Available from: GPO, Washington, DC.

[COGOS 77] Committee on Government Operations, U.S. Senate Computer Security in Federal Programs. Washington, DC: Government Printing Office; 1977 February. 298. Available from: GPO, Washington, DC.

[COLEG 78] Cole, G. D.; Heinrich, F. Design Alternatives for Computer Network Security (Vol. 1). The Network Security Center: A System Level Approach to Computer Network Security (Vol. 2). Washington, DC: GPO: 1978 January; NBSSP 500-21. Available from: GPO, Washington, DC; SN 003-003-01881-3.

[CONWR 72] Conway, R. W., Maxwell, W. L., Morgan, H. L. On the Implementation of Security Measures in Information Systems. Communications of the ACM. 15(4); 211-220; 1972 April.

[DAHLO 72] Dahl, O. J.; Dijkstra, E. W.; Hoare, C. A. R. Structured Programming. New York, NY: Academic Press Inc.; 1972. ISBN: 0-12-200550-3.

[DATAM 75] Datamation. Computer Security: Each Case is Different. Datamation. 107-109; 1975 April.

[DAVIG 78] Davida, G.; Linton, D.; Szelag, R.; Wells, D. Database Security. IEEE Transactions on Software Engineering. 1978 November.

[DENND 75] Denning, D. Secure Information Flow in Computer Systems. Ph. D. Dissertation. Purdue University; 1975.

[DENND 76] Denning, D. E. A Lattice Model of Secure Information Flow. Communications of the ACM. 19(5): 236-243; 1976 May.

[DENND 77] Denning, D. E., and Denning, P. J. Certification of Programs for Secure Information Flow. Communications of the ACM. 20(7): 4504-4513; 1977 July.

[DENND 78] Denning, D. Are Statistical Data Bases Secure? AFIPS Conference Proceedings; 1978. NCC; 1978; 47: 525-530.

[DENND 79A] Denning, D. E. Secure Personal Computing in an Insecure Network. Communications of the ACM. 22(8): 476-482; 1979 August.

[DENND 79B] Denning, D. E. and Denning, P. J. Data Security. Computing Surveys. 11(3): 227-249; 1979 September.

[DODDI 28] Department of Defense. Security Requirements for Automatic Data Processing Systems. DOD Directive 5200.28.

[DODDI 28M] Department of Defense. Techniques and Procedures for Implementing, Deactivating, Testing, and Evaluating Secure Resource Sharing ADP Systems. DOD Directive 5200.28-M.

[DODKS 78] Department of Defense Kernelized Secure Operating System. Secure Minicomputer Operating System (KSOS), Computer Program Development Specification (Type B-5). Ford Aerospace and Communications Corporation; 1978 September. WDL-7932.

[EDPAF 77] EDP Auditors Foundation for Education and Research. Control Objectives. Hanover Park, IL: EDP Auditors Foundation; 1977.

[EICHB 77] Eichman, B. A Bibliography on Privacy. The Privacy Report—ACLU. 5(2): 1-15; 1977 September.

[FABRR 74] Fabry, R. S. Capability Based Addressing. Communications of the ACM. 17(7): 403-412; 1974 July.

[FAGAM 76A] Fagan, M. E. Design and Code Inspections to Reduce Errors in Program Development. IBM Systems Journal. 15(3): 152-211; 1976.

[FAGAM 76B] Fagan, M. Design and Code Inspections and Process Control in the Development of Programs. IBM; 1976 June. TR 00.2763.

[FARRM 72] Farr, M.; Chadwick, B.; Wong, K. Security for Computer Systems. Manchester, England: National Computing Center Ltd.; 1972. 172.

[FDICO 77] Federal Deposit Insurance Company. A Guide to EDP and EFT Security Based on Occupations. Washington, DC; 1977 November.

[FEIER 79] Feiertag, Richard J.; Neumann, Peter G. The Foundations of a Provably Secure Operating System (PSOS). AFIPS Conference Proceedings, Vol. 48; 1979 June 4-7; New York, NY. Montvale, NJ: AFIPS Press; 1979: 329-334.

[FEISH 73] Feistel, H. Cryptography and Computer Privacy. Scientific American. 228(5): 15-24; 1973 May.

[FERNE 75A] Fernandez, E. B.; Summers, R. C.; Coleman, C. D. An Authorization Model for a Shared Data Base. Proceedings 1975 ACM SIGMOD International Conference; 1975.

[FERNE 75B] Fernandez, E. B.; Summers, R. C.; Lang, T. Definition and Evaluation of Access Rules in Data Management Systems. Proceedings of International Conference on Very Large Data Bases; 1975; Framingham, MA.

[FIFED 77] Fife, D. W. Computer Software Management: A Primer for Project Management and Quality Control. Washington, DC: GPO; 1977 July; NBSSP 500-11. Available from: GPO, Washington, DC; SN 003-003-01795-7.

[FIPSP 74] NBS Guidelines for Automatic Data Processing Physical Security and Risk Management. Washington, DC: GPO; 1974 June; FIPS 31. Available from: NTIS, Springfield, VA; FIPS-PUB-31.

[FIPSP 75] NBS. Computer Security Guidelines for Implementing the Privacy Act of 1974. Washington, DC: GPO; 1975 May 30; FIPS 41. Available from: NTIS, Springfield, VA; FIPS-PUB-41.

[FIPSP 76A] NBS. Guidelines for Documentation of Computer Programs and Automated Data Systems. Washington, DC: GPO; 1976 February 15; FIPS 38. Available from: NTIS, Springfield, VA; FIPS-PUB-38.

[FIPSP 76B] NBS. Glossary for Computer Systems Security. Washington, DC: GPO; 1976 February 15; FIPS 39. Available from: NTIS, Springfield, VA; FIPS-PUB-39.

[FIPSP 76C] NBS. Guidelines for Optical Character Recognition Forms. Washington, DC: GPO; 1976 May 1; FIPS 40. Available from: NTIS, Springfield, VA; FIPS-PUB-40.

[FIPSP 77A] NBS. Data Encryption Standard. Washington, DC: GPO; 1977 January 15; FIPS 46. Available from: NTIS, Springfield, VA; FIPS-PUB-46.

[FIPSP 77B] NBS. Guidelines for Evaluation of Techniques for Automated Personal Identification. Washington, DC: GPO; 1977 April 1; FIPS 48. Available from: NTIS, Springfield, VA; FIPS-PUB-48.

[FIPSP 79A] NBS. Guidelines for Documentation of Computer Programs and Automated Data Systems for the Initiation Phase; Washington, DC: GPO; 1979 August 1; FIPS 64. Available from: NTIS, Springfield, VA; FIPS-PUB-64.

[FIPSP 79B] NBS. Guidelines for Automated Data Processing Risk Analysis. Washington, DC: GPO; 1979 August 1; FIPS 65. Available from: NTIS, Springfield, VA; FIPS-PUB-65.

[FIPSP 80A] NBS. Guideline for Planning and Using a Data Dictionary System. Washington, DC: GPO; 1980 August 20; FIPS 76. Available from NTIS, Springfield, VA; FIPS-PUB-76.

[FIPSP 80B] NBS. Guideline for Planning and Management of Database Applications. Washington, DC: GPO; 1980 September 1; FIPS 77. Available from NTIS, Springfield, VA; FIPS-PUB-77.

[FITZJ 78] FitzGerald, Jerry. Internal Controls for Computerized Systems. Redwood City, CA: Jerry FitzGerald & Associates; 1978.

[FONGE 77] Fong, E. A Data Base Management Approach to Privacy Act Compliance. Washington, DC: GPO; 1977 June; NBSSP 500-10. Available from: GPO, Washington, DC; SN 003-003-01787-6.

[GAITJ 77] Gait, J. Validating the Correctness of Hardware Implementations of the NBS Data Encryption Standard. Washington, DC: GPO; 1977 November; NBSSP 500-20. Available from: GPO, Washington, DC; SN 003-003-01861-9.

[GILBT 77] Gilb, T. Software Metrics. Cambridge, MA: Winthrop Computer Systems Series; 1977.

[GLASS 77] Glaseman, S.; Turn, R.; Gaines, R. S. Problem Areas in Computer Security Assessment. AFIPS Conference Proceedings; National Computer Conference; 1977; (46): 105-112.

[GOLDB 79] Gold, B. D.; Linde, R. R.; Peeler, R. J.; Schaefer, M.; Scheid, J. F.; Ward, P. D. A Security Retrofit of VM/370. AFIPS Conference Proceedings, Vol. 48; 1979 June 4-7; New York, NY. Montvale, NJ: AFIPS Press; 1979: 335-344.

[GRAHG 72] Graham, G. S.; Denning, P. J. Protection—Principles and Practice. AFIPS Conference Proceedings; 1972; 40: 417-429.

[HANTS 76] Hantler, S. L.; King, J. C. An Introduction to Proving the Correctness of Programs. Computing Surveys. 8(3): 331-353; 1976 December.

[HEINF 76] Heinrich, F. R.; Kaufman, D. J. A Centralized Approach to Computer Network Security. AFIPS Conference Proceedings; 1976. National Computer Conference; 1976; 45: 85-90.

[HEINF 78] Heinrich, F. The Network Security Center: A System Level Approach to Computer Network Security. Washington, DC: GPO; 1978 January; NBSSP 500-21, Vol. 2. Available from: GPO, Washington, DC; SN 003-003-01881-3.

[HEMPC 73] Hemphill, C. F., Jr.; Hemphill, J. M. Security Procedures for Computer Systems. Homewood, IL: Dow Jones-Irwin; 1973.

[HOFFL 69] Hoffman, Lance J. Computers and Privacy: A Survey. Computing Surveys. 1(2): 85-103; 1969 June.

[HOFFL 70A] Hoffman, Lance J.; Miller, W. Getting a Personal Dossier from a Statistical Data Bank. Datamation. 16: 74-75; 1970 May.

[HOFFL 70B] Hoffman, Lance J. The Formulary Model for Access Control and Privacy. 1970 May. Stanford Linear Accelerator Center Report No. 117.

[HOFFL 71] Hoffman, Lance J. The Formulary Method for Flexible Privacy and Access Control. Proceedings 1971 FJCC. 1971: 587-601.

[HOFFL 73] Hoffman, Lance J., ed. Security and Privacy in Computer Systems. Los Angeles, CA: Melville Publishing Co.; 1973. ISBN: 0-471-40611-2.

[HOFFL 77] Hoffman, Lance J. Modern Methods for Computer Security and Privacy. Englewood Cliffs, NJ: Prentice-Hall, Inc.; 1977. 255. ISBN: 0-13-595207-7.

[HSIAD 74] Hsiao, D.; Kerr, D.; Stahl, F. Research on Data Secure Systems. AFIPS Conference Proceedings. NCC; 1974; 43: 994-996.

[HSIAD 79] Hsiao, David K.; Kerr, Douglas S.; Madnick, Stuart E. Computer Security. New York, NY: Academic Press; 1979. 299 pp. ISBN: 0-12-357650-4.

[HUNTM 74] Hunt, M. K.; Turn, R. Privacy and Security in Databank Systems: An Annotated Bibliography, 1970-1973. Rand Corporation Report; 1974 March; R-1361-NSF.

[HUSKJ 76] Huskamp, J. C. A Partially Annotated Bibliography for Computer Protection and Related Topics. Livermore, CA: Lawrence Livermore Laboratory; 1976 July; UCID-17198.

[IBMCO 75] International Business Machines Corporation (IBM). An Executive's Guide to Data Security—A Translation from an IBM Svenska AB Publication. New York, NY: IBM World Trade Corporation; 1975 October.

[IBMCO 76] International Business Machines Corporation (IBM). Data Security Controls and Procedures—A Philosophy for DP Installations. New York, NY: IBM World Trade Corporation; 1976 February. G320-5649-00.

[ICST 77] Institute for Computer Sciences and Technology. A Survey of Eleven Government-Developed Data Element Dictionary/Directory Systems. Washington, DC: GPO; 1977 August; NBSSP 500-16. Available from: GPO, Washington, DC; SN 003-003-01817-1.

[IIASA 77] Institute of Internal Auditors. Systems Auditability and Control Study. Altamonte Springs, FL; 1977.

[JANCE 74] Jancura, Elise G. Audit and Control of Computer Systems. New York, NY: Petrocelli/Charter; 1974. ISBN: 0-88405-283-4.

[JEFFS 76] Jeffery, S. Data Integrity and Security in the EFT Interchange Environment. Testimony, National Commission on Electronic Funds Transfer; 1976 December 16; Washington, DC. National Bureau of Standards; 33.

[JONEA 73] Jones, A. K. Protection in Programmed Systems. Ph. D. Dissertation. Carnegie-Mellon University; 1973.

[JONEA 75A] Jones, Anita K.; Lipton, Richard J. The Enforcement of Security Policies for Computation. Communications of the ACM. 197-206; 1975.

[JONEA 75B] Jones, Anita K.; Wulf, William A. Towards the Design of Secure Systems. Software—Practice and Experience. 5(4): 321-336; 1975 October-December.

[JONEA 78] Jones, A. K.; Liskov, B. H. A Language Extension for Expressing Constraints on Data Access, Communications of the ACM. 21(5): 358-367; 1978 May.

[KAHND 67] Kahn, D. The Codebreakers, The Story of Secret Writing. New York, NY: Macmillan; 1967.

[KARGP 74] Karger, P. A.; Schell, R. R. Multics Security Evaluation: Vulnerability Analysis. Hanscom AFB, MA: USAF Electronic Systems Division; 1974 June. ESD-TR-74-193, Vol. II. Available from: NTIS, Springfield, VA; AD A001120.

[KATZH 73] Katzan, H., Jr. Computer Data Security. New York, NY: Van Nostrand Reinhold Company; 1973. ISBN: 0-442-24258-1.

[KERNB 74] Kernighan, B.; Plauger, P. The Elements of Programming Style. McGraw-Hill; 1974.

[KOSAS 74] Kosarajo, S. R.; Ledgard, H. F. Concepts in Quality Software Design. Washington, DC: National Bureau of Standards; 1974 August. NBS Technical Note 842. Available from: NBS, Washington, DC; NBSTN 842.

[KRAUL 72] Krauss, Leonard I. SAFE: Security Audit and Field Evaluation for Computer Facilities and Information Systems. New Jersey: Firebrand, Krauss and Company; 1972.

[KRAUL 79] Krauss, Leonard I.; MacGahan, Aileen. Computer Fraud and Countermeasures. Englewood Cliffs, NJ: Prentice-Hall, Inc.; 1979. 509. ISBN:0-13-164772-5.

[LAMPB 71] Lampson, B. W. Protection. Proceedings of Fifth Princeton Symposium on Information Sciences and Systems; 1971 March; Princeton University. 437-443.

[LAMPB 73] Lampson, B. W. A Note on the Confinement Problem. Communications of the ACM. 16: 613-614; 1973 October.

[LAMPB 77A] Lampson, B. W.; Needham, R. M.; Randall, B.; Schroeder, M. D. Protection, Security, Reliability. Operating Systems Review. 2(1): 12-14; 1977 January.

[LAMPB 77B] Lampson, B., et al. Report on the Programming Language EUCLID. SIGPLAN Notices. 12(2); 1977 February.

[LARSR 75] Larson, R. Test Plan and Test Case Inspection Specification. IBM; 1975 April. TR 21.586.

[LEONB 77] Leong-Hong, B.; Marron, B. Technical Profile of Seven Data Element Dictionary/Directory Systems. Washington, DC: GPO; 1977 February; NBSSP 500-3. Available from: GPO, Washington, DC; SN 003-003-01725-6.

[LEWIH 80] Lewis, Harold W. The Safety of Fission Reactors. Scientific American. 242(3): 53-65; 1980 March.

[Note: This article provides an interesting discussion of risk assessment with respect to Three Mile Island nuclear reactor safety.]

[LINDR 75] Linde, R. R. Operating System Penetration. Proceedings of AFIPS 1975 National Computer Conference; 44: 361-368.

[LINDT 76A] Linden, T. A. The Use of Abstract Data Types to Simplify Program Modifications. SIGPLAN Notices. 8(2): 12-23; 1976 March.

[LINDT 76B] Linden, T., ed. Security Analysis and Enhancements of Computer Operating Systems. Washington, DC: NBS; 1976 April; NBSIR 76-1041. Available from: NTIS, Springfield, VA; PB 257 087.

[LINDT 76C] Linden, T. Operating System Structures to Support Security and Reliable Software. Washington, DC: GPO; 1976 August; NBSTN 919. Available from: GPO, Washington, DC; SN 003-003-01658-6.

[LIPNS 74] Lipner, S. B. A Panel Session—Security Kernels. AFIPS Conference Proceedings; 1974. NCC; 1974; 43: 973-980.

[LISKB 73] Liskov, B. H. Guidelines for the Design and Implementation of Reliable Software Systems. MITRE; 1973 February. MITRE Report 2345.

[LISKB 75] Liskov, B. Data Types and Program Correctness. SIGPLAN Notices; 1975 July.

[LOSOT 79] Losonsky, Terrance M. Automated Information System Security (AISS): A Comparative Analysis of Risk Management Procedures. Florida: Florida State University; 1979 December.

[LYONG 74] Lyon, G. A FORTRAN Analyzer. Washington, DC: GPO; 1974 October. NBS Technical Note (NBSTN) 849. Available from: NTIS, Springfield, VA; COM 74-50998, MF $3.00.

[MAIRW 76] Mair, W. C.; Wood, D. R.; Davis, K. W. Computer Control and Audit. Altamonte Springs, FL: Institute of Internal Auditors; 1976.

[MARTJ 73] Martin, J. Security, Accuracy, and Privacy in Computer Systems. Englewood Cliffs, NJ: Prentice-Hall; 1973.

[MCCAE 79] McCauley, E. J.; Drongowski, P. J. KSOS—The Design of a Secure Operating System. AFIPS Conference Proceedings, Vol. 48; 1979 June 4-7; New York, NY. Montvale, NJ: AFIPS Press; 1979; 345-354.

[MCCAJ 77A] McCall, Jim A.; Richards, Paul K.; Waltes, Gene F. Factors in Software Quality—Concepts and Definitions of Software Quality. Hanscom AFB, MA: United States Air Force; 1977 November; Rome Air Development Center (RADC)-TR-77-369, Vol. I. 158. Available from: NTIS, Springfield, VA; AD-A049-014.

[Note: This document contains an extensive bibliography on nearly all aspects of software development and quality.]

[MCCAJ 77B] McCall, Jim A.; Richards, Paul K.; Walters, Gene F. Factors in Software Quality—Metric Data Collection and Validation. Hanscom AFB, MA: United States Air Force; 1977 November; Rome Air Development Center (RADC)-TR-77-369, Vol. II. 145. Available from: NTIS, Springfield, VA; AD-A049-015.

[MCCAJ 77C] McCall, Jim A.; Richards, Paul K.; Walters, Gene F. Factors in Software Quality—Preliminary Handbook on Software Quality for an Acquisition Manager. Hanscom AFB, MA: United States Air Force; 1977 November; Rome Air Development Center (RADC)-TR-77-369, Vol. III. Available from: NTIS, Springfield, VA; AD-A049-055.

[MEISP 76] Meissner, P., ed. Report on the Workshop on Estimation of Significant Advances in Computer Technology. Washington, DC: NBS; 1976 December; NBSIR 76-1189. Available from: NTIS, Springfield, VA; PB 279 373.

[MILLH 77] Mills, Harlan D. Computer Science and Engineering Research Study (COSERS). Software Methodology Panel Report; 1977 March.

[MOORG 77] Moore, G. B.; Kuhns, J. L.; Treffzs, J. L.; Montgomery, C. A. Accessing Individual Records from Personal Data Files Using Nonunique Identifiers. Washington, DC: GPO; 1977 February; NBSSP 500-2. Available from: GPO, Washington, DC; SN 003-003-01726-4.

[MORRR 78] Morris, R.; Thompson, K. Password Security: A Case History. Murray Hill, NJ: Bell Laboratories; 1978 April. CSTR-71.

[MURRW 77] Murray, William H.; Wilkins, Barry. Report of the Working Group on Administrative and Physical Controls Consensus Report. Audit and Evaluation of Computer Security; 1977 March 22-24; Miami Beach, FL. Washington, DC: Government Printing Office; 1977; 256. Available from: GPO, Washington, DC; SN C13.10:500-19.

[MYERG 76] Myers, G. J. Software Reliability: Principles and Practices. New York, NY: John Wiley & Sons; 1976.

[NCEFT 77] National Commission on Electronic Funds Transfer. EFT and the Public Interest. Washington, DC: U.S. Government Printing Office; 1977 October; Final Report. Available from: GPO, Washington, DC.

[NBSIR 75] National Bureau of Standards (NBS). Index of Automated System Design Requirements as Derived from the OMB Privacy Act Implementation Guidelines. Washington, DC: NBS; 1975 October; National Bureau of Standards Internal Report (NBSIR) 75-909. Available from: NTIS, Springfield, VA; PB 246 863.

[NEUMA 77] Neumann, A. J. Features of Seven Audit Software Packages—Principles and Capabilities. Washington, DC: GPO; 1977 July; NBSSP 500-13. Available from: GPO, Washington, DC; SN 003-003-01807-4.

[NEUMP 77A] Neumann, P. G. Computer System Security Evaluation. Proceedings of AFIPS 1977 National Computer Conference. 46: 1087-1095.

[NEUMP 77B] Neumann, P. G., et al. A Provably Secure Operating System: The System, Its Applications, and Proofs. Menlo Park, CA: SRI International; 1977 February. Final Report, Project 4332.

[NEUMP 78] Neumann, P. G. A Position Paper on Attaining Secure Systems. Proceedings of U.S. Army Automation Security Workshop; 1978 December; Leesburg, VA. 26-39.

[NIBAG 79] Nibaldi, G. H. Proposed Technical Evaluation Criteria for Trusted Computer Systems. Bedford, MA: MITRE Corporation; 1979 October 25. MITRE Report No. M79-225.

[NIELN 76A] Nielsen, N. R.; Ruder, B.; Brandin, D. H. Effective Safeguards for Computer System Integrity. Proceedings of AFIPS National Computer Conference. Montvale, NJ: AFIPS Press; 1976; 45: 75-84.

[NIELN 76B] Nielsen, N. R., et al. Computer System Integrity Safeguards—System Integrity Maintenance. NSF-SRI Project 4059; 1976 October. Grant No. DCR 74-23774.

[OGDIJ 72] Ogdin, J. L. Designing Reliable Software. Datamation; 1972 July.

[OMB-C 78] Office of Management and Budget. Security of Federal Automated Information Systems. Washington, DC: OMB; 1978 July. Circular A-71, Transmittal Memorandum Number 1.

[OMB-R 79] Office of Management and Budget. Computer Security. Washington, DC: OMB; 1979 February. Available from: NTIS, Springfield, VA; PB 290 896.

[ORCEM 78] Orceyre, M. J.; Courtney, R. H., Jr. Considerations in the Selection of Security Measures for Automatic Data Processing Systems. Washington, DC: GPO; 1978 June; NBSSP 500-33. Available from: GPO, Washington, DC; SN 003-003-01946-1.

[PARKD 73] Parker, D. B., et al. Computer Abuse. Palo Alto, CA: Stanford Research Institute; 1973 November.

[PARKD 75] Parker, D. B. Computer Abuse Assessment. Menlo Park, CA: Stanford Research Institute; 1975 December.

[PARKD 76] Parker, D. B. Crime by Computer. New York, NY: Charles Scribner & Sons; 1976.

[PATRR 78] Patrick, R. L. Performance Assurance and Data Integrity Practices. Washington, DC: GPO; 1978 January; NBSSP 500-24. Available from: GPO, Washington, DC; SN 003-003-01879-1.

[PFISJ 76] Pfister, J. J. An Annotated and Cross-Referenced Bibliography on Computer Security and Privacy (1973-1975). College Station, TX: Industrial Engineering Department of Texas A. & M. University; 1976 March; Masters thesis.

[PMPFA 75] National Bureau of Standards, MITRE Corporation, The Privacy Mandate—Planning for Action. Symposium/Workshop; 1975 April 2-4; Washington, DC. 1975: 102.

[POPEG 74] Popek, G. J. Protection Structures. Computer. 7(6): 22-31; 1974 June.

[POPEG 78A] Popek, G. J.; Kline, C. S. Issues in Kernel Design. Proceedings of AFIPS National Computer Conference. Montvale, NJ: AFIPS Press; 1978; 47: 1079-1086.

[POPEG 78B] Popek, G. J.; Farber, D. A. A Model for Verification of Data Security in Operating Systems. Communications of the ACM. 21(9): 737-749; 1978 September.

[POPEG 79A] Popek, G. J.; Kline, C. Encryption and Secure Computer Networks. Computing Surveys. 11: 331-356; 1979.

[POPEG 79B] Popek, G. J., et al. UCLA Secure Unix. Proceedings of the 1979 National Computer Conference. 1979 June; 563-572.

[PORTW 77] Porter, W. T.; Perry, W. E. EDP Controls and Auditing. Belmont, CA: Wadsworth Publishing Company; 1977. Second Edition.

[PURDG 74] Purdy, G. A High Security Log-In Procedure. Communications of the ACM. 17: 442-445; 1974 August.

[PWCO 79] Price Waterhouse & Company. Guide to Accounting Controls. New York, NY: Price Waterhouse & Co.; 1979. PW 946001-9.

[RCBCG 76A] U.S. General Accounting Office. Report to Congress by the Comptroller General of the United States: Improvements Needed in Managing Automated Decisionmaking by Computers Throughout the Federal Government. Washington, DC: U.S. General Accounting Office; 1976 April 23. FGMSD-76-5. Available from: GAO, Washington, DC.

[RCBCG 76B] U.S. General Accounting Office. Report to Congress by the Comptroller General of the United States: Computer-Related Crimes in Federal Programs. Washington, DC: U.S. General Accounting Office; 1976 April 27. FGMSD-76-27. Available from: GAO, Washington, DC.

[RCBCG 76C] U.S. General Accounting Office. Report to Congress by the Comptroller General of the United States: Managers Need to Provide Better Protection for Federal Automatic Data Processing Facilities. Washington, DC: U.S. General Accounting Office; 1976 May 10. FGMSD-76-40. Available from: GAO, Washington, DC.

[RCBCG 77] U.S. General Accounting Office. Report to Congress by the Comptroller General of the United States: New Methods Needed for Checking Payments Made by Computers. Washington, DC: U.S. General Accounting Office; 1977 November 7. FGMSD-76-82. Available from: GAO, Washington, DC.

[RCBCG 78A] U.S. General Accounting Office. Report to Congress by the Comptroller General of the United States: Challenges of Protecting Personal Information in an Expanding Federal Computer Network Environment. Washington, DC, U.S. General Accounting Office; 1978 April 28. LCD-76-102. Available from: GAO, Washington, DC.

[RCBCG 78B] U.S. General Accounting Office. Report to Congress by the Comptroller General of the United States: Strong Centralized Management Needed in Computer-Based Information Systems. Washington, DC: U.S. General Accounting Office; 1978 May 22. LCD-78-105. Available from: GAO, Washington, DC.

[RCBCG 78C] U.S. General Accounting Office. Report to Congress by the Comptroller General of the United States: New Ways of Preparing Data for Computers Could Save Money and Time and Reduce Errors. Washington, DC: U.S. General Accounting Office; 1978 July 18. FGMSD-78-39. Available from: GAO, Washington, DC.

[RCBCG 79] U.S. General Accounting Office. Report to Congress by the Comptroller General of the United States: Automated Systems Security—Federal Agencies Should Strengthen Safeguards Over Personal and Other Sensitive Data. Washington, DC: U.S. General Accounting Office; 1979 January 23. LCD-78-123. Available from: GAO, Washington, DC.

[REEDS 73] Reed, S. K.; Gray, M. Controlled Accessibility Bibliography. Washington, DC: GPO; 1973 June; National Bureau of Standards Technical Note (NBSTN) 780. Available from: NTIS, Springfield, VA; COM 73-50533.

[REEDS 74] Reed, S. K.; Branstad, D., eds. Controlled Accessibility Workshop Report. Washington, DC: GPO; 1974 May; NBSTN 827. Available from: NTIS, Springfield, VA; COM 74-50457.

[REIMG 77] Reimherr, G. W. Computer Information Security and Protection. Washington, DC: NTIS; 1964-1977 June; Vol. 1. Available from: NTIS, Springfield, VA; PS-78/0859.

[REIMG 78A] Reimherr, G. W. Computer Information Security and Protection. Washington, DC: NTIS; 1977 July-1978 May; Vol. 2. Available from: NTIS, Springfield, VA; PS-78/0860.

[REIMG 78B] Reimherr, G. W. Computer Information Security and Protection (Citations from the Engineering Index Data Base). Washington, DC: NTIS; 1978 August. Available from: NTIS, Springfield, VA; PS-78/0861.

[RENNC 74] Renninger, C.; Branstad, D., ed. Government Looks at Privacy and Security in Computer Systems. Washington, DC: GPO; 1974 February; NBSTN 809. Available from: NTIS, Springfield, VA; COM 74-50174.

[RENNC 74A] Renninger, C., ed. Approaches to Privacy and Security in Computer Systems. Washington, DC: Government Printing Office (GPO); 1974 September; National Bureau of Standards Special Publication (NBSSP) 404. Available from: GPO, Washington, DC; SN 003-003-01319-6.

[RIVER 78] Rivest, R.; Shamir, A.; Adleman, L. A Method for Obtaining Digital Signatures and Public Key Cryptosystems. Communications of the ACM. 21: 120-126; 1978.

[RUDEB 78] Ruder, B.; Madden, J. D. An Analysis of Computer Security Safeguards for Detecting and Preventing Intentional Computer Misuse. Washington, DC: GPO;1978 January;NBSSP 500-25. Available from: GPO, Washington, DC; SN 003-003-01871-6.

[RUTHZ 77] Ruthberg, Z.; McKenzie, R., ed. Audit and Evaluation of Computer Security. Washington, DC: GPO; 1977 October; National Bureau of Standards Special Publication (NBSSP) 500-19. Available from: GPO, Washington, DC; SN 003-003-01848-1.

[RUTHZ 78] Ruthberg, Z., ed. Audit and Evaluation of Computer Security II: System Vulnerabilities and Controls. Washington, DC: GPO; 1978 November 28-30; National Bureau of Standards Special Publication (NBSSP) 500-57. Available from: GPO, Washington, DC; SN 003-003-02178-4.

[SALTJ 74] Saltzer, J. Ongoing Research and Development on Information Protection. ACM Operating Systems Review. 8(3): 8-24; 1974 July.

[SALTJ 75] Saltzer, J. H.; Schroeder, M. D. The Protection of Information in Computer Systems. Proceedings of the IEEE. Institute of Electrical and Electronics Engineers, Inc.; 1975; 63(9): 1278-1308.

[SCHEJ 74] Scherf, J. A. Computer and Data Base Security: A Comprehensive Annotated Bibliography. Cambridge, MA: Massachusetts Institute of Technology Project MAC; 1974 January; MAC TR-122.

[SCHIW 75] Schiller, W. L. The Design and Specification of a Security Kernel for the PDP-11/45. Bedford, MA: The MITRE Corporation; 1975 May. ESD-TR-75-69. Available from: NTIS, Springfield, VA; AD A011-712.

[SCHRM 77] Schroeder, M.; Clark, D.; Slatzer, J. The MULTICS Kernel Design. Proceedings of the Sixth Symposium on Operating Systems Principles; 1977 November; West Lafayette, IN.

[SCOML 75] Scoma, Louis, Jr. Data Center Security. Data Management. 13: 19-21; 1975 September.

[SGCCA 75A] Study Group on Computer Control and Audit Guidelines. Computer Audit Guidelines. Toronto, Canada: Canadian Institute of Chartered Public Accountants; 1975.

[SGCCA 75B] Study Group on Computer Control and Audit Guidelines. Computer Control Guidelines. Toronto, Canada: Canadian Institute of Chartered Public Accountants; 1975.

[SHANC 49] Shannon, C. E. Communication Theory of Secrecy Systems. Bell Systems Technical Journal. 10: 656-715.

[STONM 74] Stonebraker, M.; Wong, E.; Wong, M. Access Control in a Relational Data Base Management System by Query Modification. Berkeley, CA: University of California; 1974 May. ERL-M438.

[TURNR 75] Turn, R.; Ware, W. H. Privacy and Security in Computer Systems. American Scientist. 63(2): 196-203; 1975 March-April. Available from: NTIS, Springfield, VA; AD-A016-493.

[USGAO 72] U.S. General Accounting Office. Standards for Audit of Governmental Organizations, Programs, Activities, and Functions. Washington, DC: Government Printing Office; 1972. Available from: GPO, Washington, DC; SN 2000-00110.

[USGAO 74] U.S. General Accounting Office. Internal Auditing in Federal Agencies. Washington, DC: General Accounting Office; 1974. Available from: GAO, Washington, DC.

[USGAO 79A] U.S. General Accounting Office. Auditing Computer-Based Systems. Washington, DC: Government Printing Office; 1979. Available from: GPO, Washington, DC; SN 020-000-00174-7.

[USGAO 79B] U.S. General Accounting Office, Financial and General Management Studies Division. Questions Designed to Aid Managers and Auditors in Assessing the ADP Planning Process. Washington, DC: U.S. General Accounting Office; 1979 August. Exposure Draft. Available from: GAO, Washington, DC.

[VAN D 72] Van Tassel, D. Computer Security Management. Englewood Cliffs, NJ: Prentice-Hall, Inc.; 1972.

[VAN D 74] Van Tassel, Dennie. Program Style, Design, Efficiency, Debugging and Testing. Englewood Cliffs, NJ: Prentice-Hall, Inc.; 1974.

[WAREW 70] Ware, W., et al. Security Controls for Computer Systems. Rand Corporation; 1970. Technical Report R-609.

[WEING 71] Weinberg, G. M. The Psychology of Computer Programming. New York, New York: Van Nostrand Reinhold; 1971.

[WEISC 73] Weissman, C. System Security Analysis/Certification Methodology and Results. 1973 October. SDC SP-3728.

[WEISH 74] Weiss, Harold. Computer Security, An Overview. Datamation. 20: 42-47; 1974 January.

[WESTA 72] Westin, A. F.; Baker, M. A. Databanks in a Free Society. New York, NY: Quadrangle Books; 1972.

[WESTA 76] Westin, A. F. Computers, Health Records, and Citizen Rights. Washington, DC: NBS; 1976 December; National Bureau of Standards Monograph (NBSM) 157. Available from: Government Printing Office (GPO), Washington, DC; SN 003-003-01681-1.

[WESTA 77] Westin, A. F.; Isbell, F. A Policy Analysis of Citizen Rights Issues in Health Data Systems. Washington, DC: GPO; 1977 January; NBSSP 469. Available from: GPO, Washington, DC; SN 003-003-01730-2.

[WESTA 79] Westin, A. F. Computers, Personnel Administration, and Citizen Rights. Washington, DC: GPO; 1979 July; NBSSP 500-50. Available from: GPO, Washington, DC; SN 003-003-02087-7.

[WIRTN 73] Wirth, Niklaus. Systematic Programming—An Introduction. Englewood Cliffs, NJ: Prentice-Hall, Inc.; 1973. ISBN: 0-13-880369-2.

[WONGK 75A] Wong, K. K. Putting a Price on Information. Computer Management; 1975 March.

[WONGK 75B] Wong, K. K. Information—How to Analyze its Value. Computer Management; 1975 April.

[WONGK 77] Wong, K. K. Computer Security Risk Analysis and Control, A Guide for the DP Manager. Manchester, England: The National Computing Centre Limited; 1977. ISBN: 0-8104-5466-1 (USA), or 0-85012-179-5 (Europe).

[WOODH 77] Wood, H. The Use of Passwords for Controlled Access to Computer Resources. Washington, DC: GPO; 1977 May; NBSSP 500-9. Available from: GPO, Washington, DC; SN 003-003-070-1.

[WOODJ 77] Woodward, J. P. L.; Nibaldi, G. H. A Kernel-Based Secure UNIX DESIGN. Bedford, MA: MITRE Corporation; 1977 November. MTR-3499.

[WOODJ 79] Woodward, John P. L. Applications for Multilevel Secure Operating Systems. AFIPS Conference Proceedings, Vol. 48; 1979 June 4-7; New York, NY. Montvale, NJ: AFIPS Press; 1979: 319-328.

[WULFW 74A] Wulf, W. A. Toward a Language to Support Structured Programs. Pittsburgh, PA: Computer Science Department, Carnegie-Mellon University; 1974 April.

[WULFW 74B] Wulf, W. A., et al. HYDRA: The Kernel of a Multiprocessor Operating System. Communications of the ACM. 17(6): 337-345; 1974 June.

[WULFW 76] Wulf, W. A.; London, R. L.; Shaw, M. Abstraction and Verification in Alphard: Introduction to Language and Methodology. Pittsburgh, PA: Carnegie-Mellon University; 1976 June; Technical Report.

[YOURE 75] Yourdon, E. Techniques of Program Structure and Design. Englewood Cliffs, NJ: Prentice-Hall, Inc.; 1975.

U.S. GOVERNMENT PRINTING OFFICE : 1981 O - 30-838